
150 ptas
mi computer
44
CURSO PRACTICO DEL ORDENADOR PERSONAL,
EL MICRO Y EL MINIORDENADOR
Editorial Delta, S.A.

MI COMPUTER CURSO PRACTICO
DEL ORDENADOR PERSONAL, EL MICRO Y EL MINIORDENADOR

Publicado por Editorial Delta, S.A., Barcelona

Volumen IV - Fascículo 44

Director: José Mas Godayol
Director editorial: Gerardo Romero
Jefe de redacción: Pablo Parra
Coordinación editorial: Jaime Mardones, Francisco Martín
Asesor técnico: Ramón Cervelló

Redactores y colaboradores: G. Jefferson, R. Ford, F. Martín, S. Tarditti, A. Cuevas, F. Blasco
Para la edición inglesa: R. Pawson (editor), D. Tebbutt (consultant editor), C. Cooper (executive editor), D. Whelan (art editor), Bunch Partworks Ltd. (proyecto y realización)

Realización gráfica: Luis F. Balaguer

Redacción y administración:
Paseo de Gracia, 88, 5.°, 08008 Barcelona
Tels. (93) 215 10 32 / (93) 215 10 50 - Télex 97848 EDLTE

MI COMPUTER, *Curso práctico del ordenador personal, el micro y el miniordenador*, se publica en forma de 96 fascículos de aparición semanal, encuadernables en ocho volúmenes. Cada fascículo consta de 20 páginas interiores y sus correspondientes cubiertas. Con el fascículo que completa cada uno de los volúmenes, se ponen a la venta las tapas para su encuadernación.
El editor se reserva el derecho de modificar el precio de venta del fascículo en el transcurso de la obra, si las circunstancias del mercado así lo exigieran.

ISBN: 84-85822-83-8 (fascículo) 84-7598-005-8 (tomo 4)
84-85822-82-X (obra completa)
Depósito Legal: B. 52/1984

Fotocomposición: Tecfa, S.A., Pedro IV, 160, Barcelona-5
Impresión: Cayfosa, Santa Perpètua de Mogoda (Barcelona) 148411
Impreso en España - Printed in Spain - Noviembre 1984

Editorial Delta, S.A., garantiza la publicación de todos los fascículos que componen esta obra.

Distribuye para España: Marco Ibérica, Distribución de Ediciones, S.A., Carretera de Irún, km 13,350. Variante de Fuencarral, 28034 Madrid.
Distribuye para Argentina: Viscontea Distribuidora, S.C.A., La Rioja 1134/56, Buenos Aires.
Distribuye para Colombia: Distribuidoras Unidas, Ltda., Transversal 93, n.° 52-03, Bogotá D.E.
Distribuye para México: Distribuidora Intermex, S.A., Lucio blanco, n.° 435, Col. San Juan Tlihuaca, Azcapotzalco, 02400, México D.F.
Distribuye para Venezuela: Distribuidora Continental, S.A., Edificio Bloque Dearmas, final Avda. San Martín con final Avda. La Paz, Caracas 1010.

Pida a su proveedor habitual que le reserve un ejemplar de MI COMPUTER. Comprando su fascículo todas las semanas y en el mismo quiosco o librería, Vd. conseguirá un servicio más rápido, pues nos permite realizar la distribución a los puntos de venta con la mayor precisión.

Servicio de suscripciones y atrasados (sólo para España)

Las condiciones de suscripción a la obra completa (96 fascículos más las tapas, guardas y transferibles para la confección de los 8 volúmenes) son las siguientes:

a) Un pago único anticipado de 16 690 ptas. o bien 8 pagos trimestrales anticipados y consecutivos de 2 087 ptas. (sin gastos de envío).

b) Los pagos pueden hacerse efectivos mediante ingreso en la cuenta 6.850.277 de la Caja Postal de Ahorros y remitiendo a continuación el resguardo o su fotocopia a Editorial Delta, S.A. (Paseo de Gracia, 88, 5.°, 08008 Barcelona), o también con talón bancario remitido a la misma dirección.

c) Se realizará un envío cada 12 semanas, compuesto de 12 fascículos y las tapas para encuadernarlos.

Los fascículos atrasados pueden adquirirse en el quiosco o librería habitual. También pueden recibirse por correo, con incremento del coste de envío, haciendo llegar su importe a Editorial Delta, S.A., en la forma establecida en el apartado b).

Para cualquier aclaración, telefonear al (93) 215 75 21.

No se efectúan envíos contra reembolso.



Escribir para la pantalla

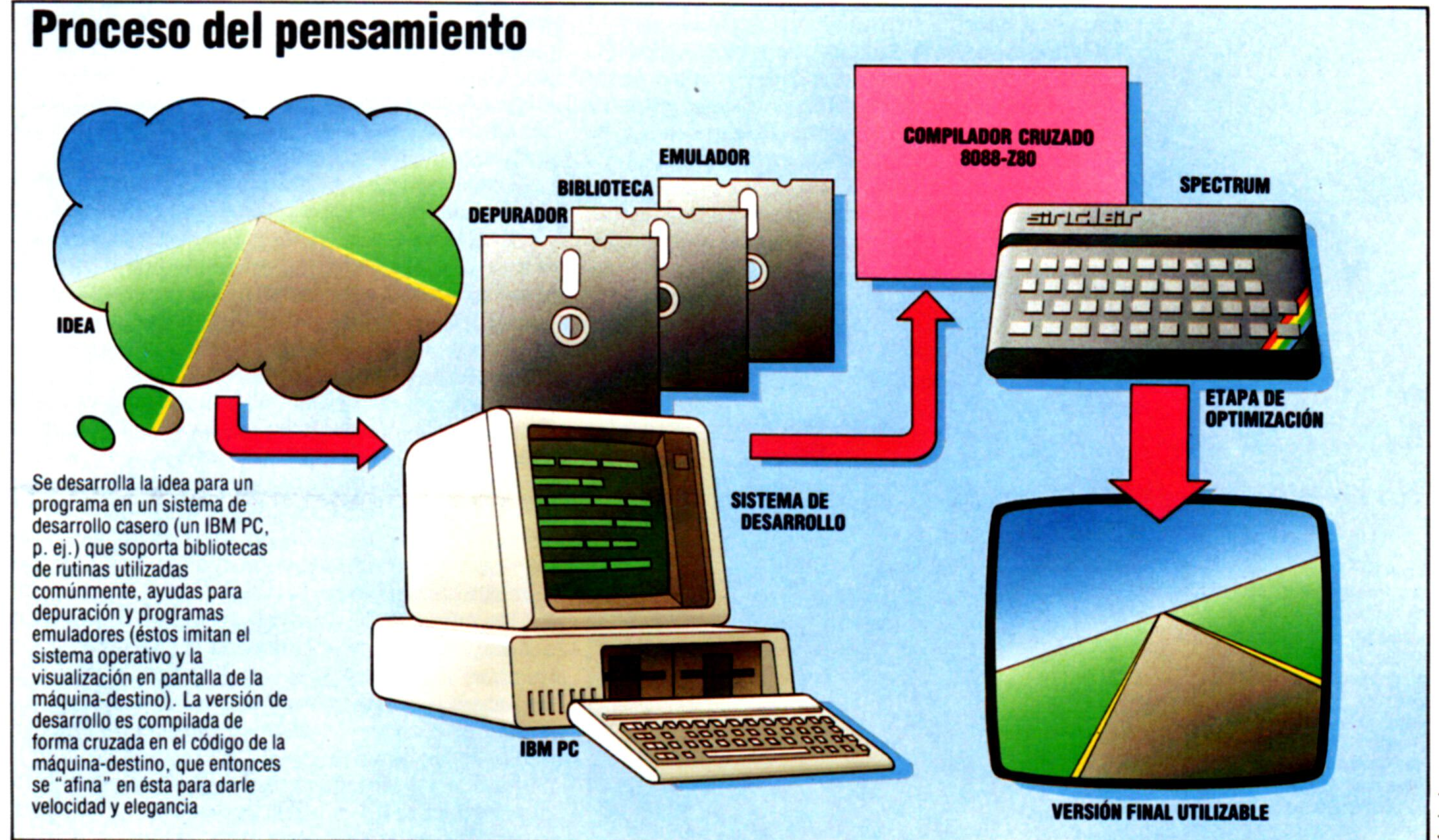
Las empresas de software profesionales utilizan un hardware muy complejo e invierten enormes recursos en la creación de sus productos

Un equipo caro no significa necesariamente programas de éxito; algunos escritores aficionados han conseguido hacerse una pequeña fortuna con software que han producido en su casa, en un Spectrum. Como quiera que sea, los niños prodigio que programan en sus hogares se están convirtiendo en una especie en vías de extinción, debido en especial al desarrollo de las grandes firmas de software durante estos últimos años.

Uno de los atributos más importantes del software serio para máquinas personales es la velocidad de operación; y ello significa que es necesario escribir los programas (al menos en parte) en código máquina. Pero trabajar con código máquina es sumamente difícil; en particular, los programadores de este lenguaje necesitan piezas de software adicionales que los ayuden a escribir sus programas. Como mínimo se requerirá un programa ensamblador para traducir el código fuente del programador al código objeto que la máquina entiende, y esta parte puede ser bastante desalentadora cuando se trata de un programa largo. Muchos escritores de software trabajan de esta manera.

Básicamente, la calidad de los programas ensambladores existentes para máquinas personales es baja. Hasta el más sencillo de estos paquetes emplea considerables cantidades de memoria y, por consiguiente, limita las dimensiones de los programas que se pueden escribir con el mismo. Asimismo, muchas máquinas personales son muy incómodas de utilizar durante períodos prolongados: los teclados pobres, las visualizaciones imperfectas y, en algunos casos, la falta de unidades de disco, hace que emplear un equipo de estas características sea una ardua tarea.

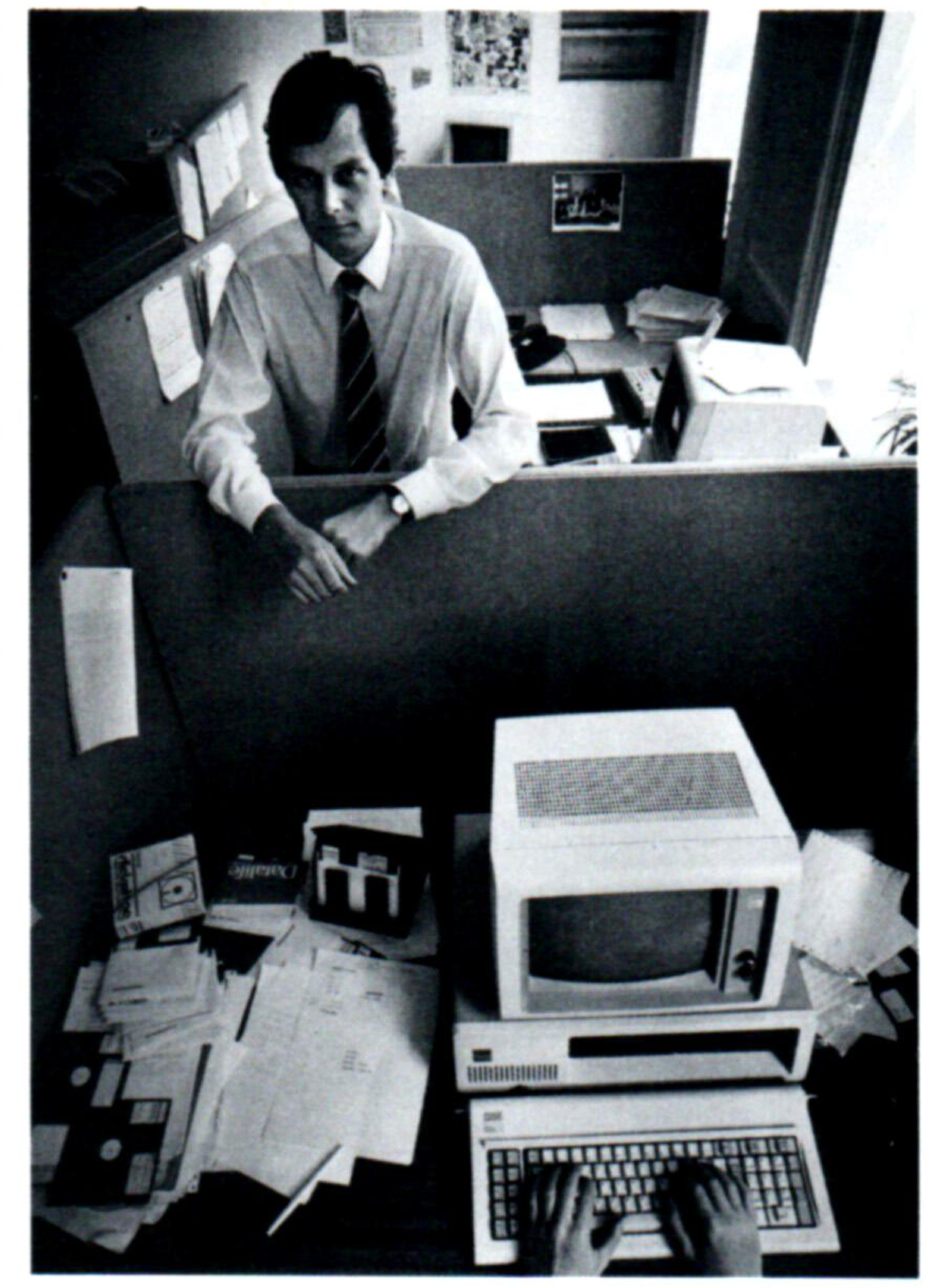
Por estas razones, la mayoría de las empresas profesionales no utilizan el micro para el cual está destinado el programa (llamado *máquina-destino*), sino que emplean ordenadores de oficina con software especial (denominados *sistemas de desarro-*



Se desarrolla la idea para un programa en un sistema de desarrollo casero (un IBM PC, p. ej.) que soporta bibliotecas de rutinas utilizadas comúnmente, ayudas para depuración y programas emuladores (éstos imitan el sistema operativo y la visualización en pantalla de la máquina-destino). La versión de desarrollo es compilada de forma cruzada en el código de la máquina-destino, que entonces se "afina" en ésta para darle velocidad y elegancia

Kevin Jones

Intelligent Software
Especializada en juegos de estrategia como el ajedrez, IS utiliza máquinas IBM y Apple con sus propias interfaces especialmente creadas para desarrollar el software. A IS, la división de programas en segmentos dependientes de la máquina y universales hace que le resulte más sencillo soportar una gama de ordenadores y máquinas exclusivas para jugar al ajedrez


Tony Sleep

llo). Los programadores que hacen uso de estas máquinas escriben con frecuencia en lenguajes tales como el PASCAL y el C. Utilizan versiones de estos lenguajes conocidas como compiladores cruzados o ensambladores cruzados, que permiten realizar el trabajo en un micro que cuenta con un procesador 8086, por ejemplo, mientras que los programas así producidos funcionarán en máquinas con procesadores Z80. Estos compiladores cruzados son lenguajes de alto nivel (como el BASIC), lo que los hace sencillos de utilizar por el programador, pero los programas que crean están escritos en código máquina. Los programadores de código máquina experimentados someten a un cuidadoso examen los programas así desarrollados y a menudo logran optimizarlos aún más.

Es evidente que un sistema de desarrollo posee una enorme ventaja respecto al micro personal. Un ensamblador basado en disco, o uno que utilice un espacio de RAM ampliado para almacenar tablas más grandes, trabajará más eficazmente que un ensamblador copiado desde una cinta y que opera en los confines de un micro personal. A la versión de desarrollo del código se le pueden agregar rutinas de depuración, sin necesidad de preocuparse por si el código generado es demasiado extenso para la memoria. Asimismo, es muchísimo mejor trabajar en un ordenador de oficina que posee un buen teclado, una visualización nítida y unidades de disco.

Una de las firmas que se valen de esta técnica de desarrollo de programas es Intelligent Software (IS), fundada en 1981 a raíz de la combinación de la experiencia de David Levy, el especialista en ajedrez, y ANT Microware, de Robert Madge. La empresa se especializa en juegos de estrategia, escritos mayormente por contrato para los micros personales populares. También desarrolla el aspecto software de las máquinas para jugar al ajedrez.

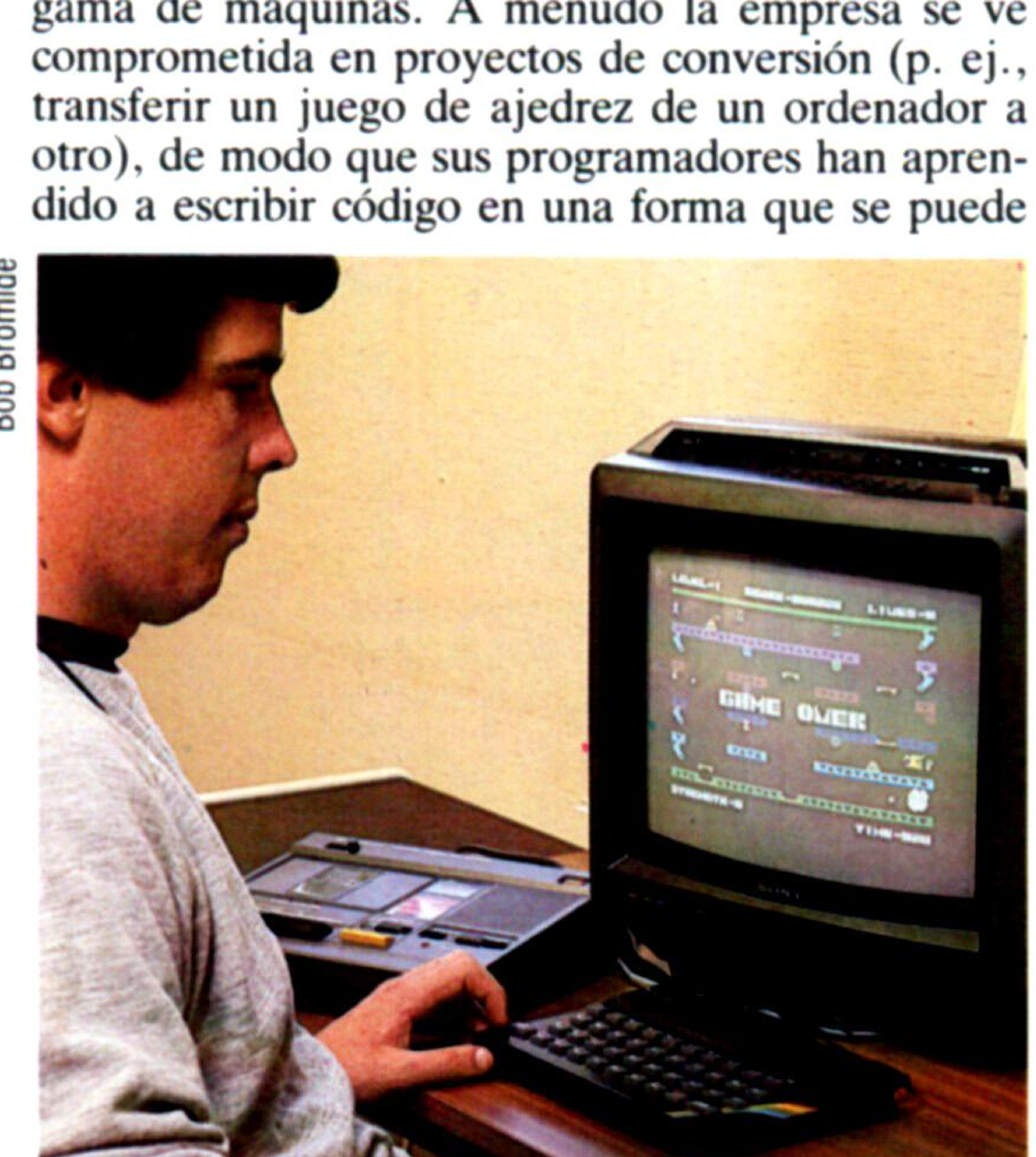
Además de utilizar para el desarrollo las propias máquinas-destino, IS emplea ordenadores IBM PC y Apple con interfaces desarrolladas especialmente para permitir el intercambio de código entre su gama de máquinas. A menudo la empresa se ve comprometida en proyectos de conversión (p. ej., transferir un juego de ajedrez de un ordenador a otro), de modo que sus programadores han aprendido a escribir código en una forma que se puede segmentar fácilmente. Un nivel de segmentación que resulta útil cuando hay que pasar código de un procesador a otro es la división del programa en código de juego y código de entrada-salida. En la nueva máquina, el código de E/S tendrá diferentes direcciones de puertas o de memoria y tal vez también sea distinto desde el punto de vista estratégico (sondeo *—polling—* reemplazado por interrupciones, etc.). Quizá se requiera cierto ingenio para soslayar las limitaciones del hardware, pero normalmente no habrá una cantidad excesiva de entrada-salida. Por el contrario, habrá código de juego en abundancia, pero como éste está aislado del hardware (excepto, por supuesto, del procesador) su conversión será directa.

IS desea evitar las restricciones impuestas sobre la capacidad de inventiva de los programadores, de modo que las "reglas de la casa" que regulan la escritura de código son muy pocas. Un punto importante en el que ellos insisten, sin embargo, es en que el código fuente incluya numerosos comentarios, de modo que siempre resulte claro qué es lo que están haciendo las rutinas.

Cuando los programadores trabajan para una firma de software desde su casa, cada uno de ellos desarrollando su propio proyecto, no es posible usar demasiados recursos de la firma en cuestión. En este caso, la individualidad se preserva al precio de una gran cantidad de esfuerzo duplicado, porque cada programador debe reinventar el código para rutinas similares.

Una empresa de software, Psion, está utilizando ordenadores aún más grandes que el IBM PC. De las casas de software que escriben para el mercado de juegos por ordenador personal, Psion es de las pocas que lleva a cabo el grueso de su desarrollo en miniordenadores.

Psion se inició como empresa desarrollando software para el ZX81 (y utilizando equipos ZX81 para hacerlo). Cuando empezó crear la cinta Horizons




Bob Bromide

Visions
Los programadores que trabajan en sus hogares con las máquinas-destino constituyen el grueso de los esfuerzos de programación de Visions. Después de que se hayan decidido la idea del juego y sus escenarios, las rutinas componentes se desarrollan en el lenguaje assembly nativo (Z80 y 6502) utilizando ensambladores como el HiSoft Dev-Pak en el Spectrum

que se edita con cada Spectrum, Psion adquirió un TRS-80 con discos, una máquina que utiliza el mismo procesador Z80, y construyó una interface especial entre las dos máquinas. Pero en agosto de 1982, la empresa decidió que no podía continuar adoptando un sistema de desarrollo completamente diferente cada vez que salía al mercado un nuevo ordenador personal. De manera que reinvirtió los beneficios en comprar hardware pesado de muchísima potencia. En principio, estas máquinas deberían ser suficientemente flexibles para hacer frente a cualquier ordenador que pudiera deparar el futuro. Las máquinas elegidas fueron dos Vax 750, que ejecutan el sistema operativo VMS de DEC.

Las Vax 750 le reportaron a Psion dos ventajas: la calidad del software suministrado por DEC, con la oportunidad que proporciona para crear ayudas de software diseñadas especialmente, y la gran "fuerza bruta" basada en la combinación del sistema operativo y el hardware. Hay muchísimo lugar para un conjunto de ayudas al software, como compiladores, bibliotecas de subrutinas comunes y programas de depuración, todo ello compartido por los programadores, entre 16 y 20, que pueden compartir una sola máquina al mismo tiempo. Las dos máquinas permiten transferir software con toda facilidad de la una a la otra cuando es necesario.

Las bibliotecas de subrutinas comunes ya formaban parte de la filosofía de Psion en los días del TRS-80, pero en un sistema dual de discos flexibles, el intercambio de datos entre discos resultaba tedioso. Las nuevas máquinas Vax permiten que los equipos de programadores trabajen juntos, compartiendo bibliotecas de proyectos comunes de las cuales se pueden llamar módulos casi al instante, y las bibliotecas pueden incluso ser compartidas entre equipos que trabajan en proyectos diferentes. Ésta es la gran ventaja de un sistema de tiempo compartido; y con la ventaja adicional de que también se hará cargo del trabajo administrativo sin tener que interrumpir a los programadores. Psion tiene planeado agregar un tercer Vax para apoyar las tareas administrativas, dejando dos máquinas libres para la producción de software.

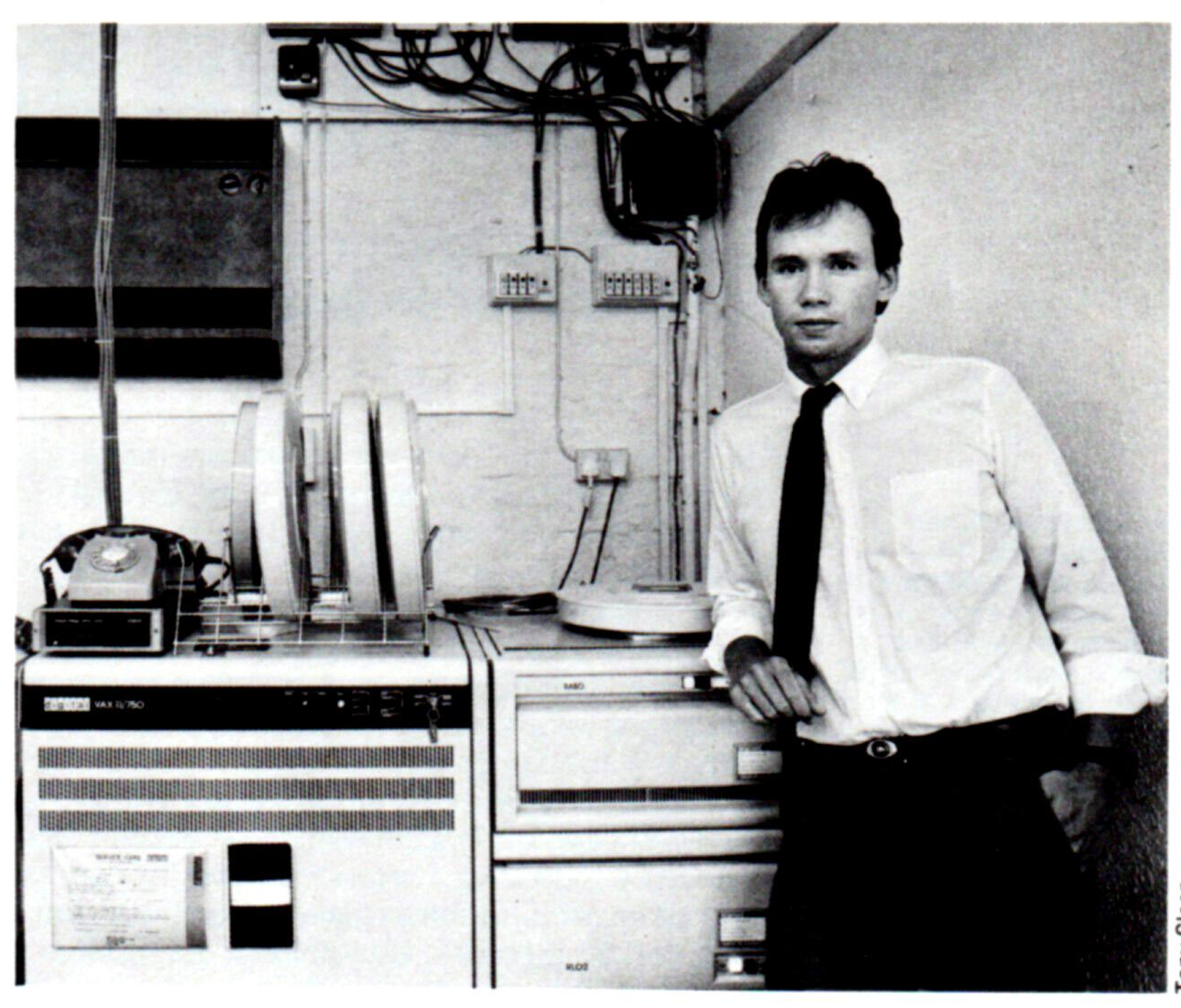
Aun cuando se lo pudiera permitir desde el punto de vista económico, estaría equivocado si creyera que sólo con salir y comprarse un Vax usted quedaría instantáneamente al nivel de Psion. De este ambiente de trabajo de Psion, tan bien desarrollado, es muy poco lo que DEC le ha servido en bandeja. Ha llevado muchísimo trabajo, duro y tenaz, hacer que las tareas sencillas se llevaran a cabo con eficacia y fiabilidad, así como lograr las ayudas al software y utilidades forjadas a mano (escritas en c) que Psion ha agregado.

Psion utiliza el c, un lenguaje de "nivel intermedio" que puede producir código objeto razonablemente rápido y compacto para chips de 16 bits como el 8086, que dista mucho del compilador c para ocho bits. De modo que al escribir para máquinas-destino como el Spectrum ha sido necesario que Psion desarrollara sus propias técnicas especiales. Psion es más bien reacia a revelar sus secretos, pero se sabe que utilizó el c para escribir su propio compilador, al cual, a falta de un nombre propio, se lo denomina "nuestro lenguaje de mesa". Éste se parece un poco al c, es portable entre distintos procesadores y crea un código muy eficaz.

Existe una regla universal según la cual el mantenimiento del sistema y la escritura de ayudas para programación caseras, como el lenguaje de mesa, por lo general representan el 30 % de todo el esfuerzo de programación; pero para Psion el tiempo extra bien vale la pena. Desarrollar el código fuente en casa significa la propiedad total: uno puede desmontarlo y mejorarlo o adaptarlo de una forma que sería absolutamente imposible en el caso del software adquirido comercialmente. Si en el software comprado aparece un error, localizarlo es difícil si no imposible y, por lo general, no existe la posibilidad de efectuar cambios internos.

El software especial adquirido por Psion incluye programas que son simulaciones exactas de microprocesadores populares, como el Z80 y el 6502. Por consiguiente, se puede hacer que los gigantescos ordenadores Vax se comporten exactamente como si fueran un Commodore 64 o un Spectrum. A pesar de la potencia de los ordenadores Vax, los simuladores trabajan a una fracción de la velocidad de la máquina destino. La ventaja es que permiten que el programador observe el contenido de todos los registros del interior del microprocesador en cualquier etapa del programa. Esto es especialmente útil para rastrear los errores de los programas. Cuando un programa en código máquina va mal y se cuelga, el programador normalmente no puede decir qué es lo que pasó. Psion puede, por lo tanto, ahorrarse muchas horas en la depuración.

Gran parte de los esfuerzos recientes de Psion en cuanto a desarrollo han estado dirigidos a producir el juego de cuatro programas estándar de gestión que se suministran con el Sinclair QL. La familia de chips Motorola 68000, uno de los cuales es el del QL, se diseñó alrededor de lenguajes de alto nivel, y los programas en c se compilan tan eficientemente en estos chips que hacen innecesaria la escritura en ensamblador. Si todos los ordenadores personales siguieran la ruta señalada por el QL, el c podría reemplazar por completo al ensamblador, y tanto Psion como las casas de software más pequeñas se olvidarían para siempre del trabajo, propio de titanes, que supone la traducción de código a mano.


Tony Sleep

Psion
En 1982 esta empresa adquirió un par de miniordenadores Vax 750 como base de su sistema de desarrollo de software. Cada máquina permite que hasta 20 programadores utilicen simultáneamente la gama de compiladores cruzados, bibliotecas de software y depuradores para crear y traducir programas

Aventura en compañía

Con el MUD muchos jugadores pueden conectar su ordenador a un ordenador central para participar en un juego colectivo

Laberinto de emociones
Las aventuras que utilizan ordenadores centrales permiten la participación de muchos jugadores; en el Multi-User Dungeon, hasta 43 Aprendices, Guerreros, Encantadoras, etc., compiten o colaboran para reunir tesoros y convertirse en Brujos o Brujas omnipotentes. Las aventuras con ordenadores centrales también ofrecen escenarios grandes y detallados: el MUD ofrece El Campo, varias cavernas, un bosque, la isla del dragón, El Mar y El Pantano, todos los cuales pueden contener tesoros y estar habitados por duendes y zombies. La conexión con los jugadores y sus micros personales se realiza mediante seis líneas telefónicas

MUD (*Multi-User Dungeon:* mazmorra multiusuario) es una aventura en tiempo real en la que usted se encuentra con otros jugadores, con los que puede mantener conversaciones, pedirles consejo, unirse contra un enemigo común o luchar contra ellos. Estos jugadores no forman parte del programa: están participando en este juego al mismo tiempo que usted, y, por lo tanto, acciones suyas influyen sobre las de usted.

El MUD se ejecuta en un ordenador DEC10 gigante instalado en la Universidad de Essex (Gran Bretaña) y todo lo que necesita el usuario para jugarlo en su propio microordenador es un programa emulador de terminal, un teléfono, un modem y una cuenta PSS (*Packet Switching System*: sistema de conmutación de paquetes). Un programa emulador de terminal permite que su micro se comunique con el ordenador central a través de un enlace telefónico. El usuario puede escoger entre escribir su propio emulador o comprarse uno. Lo ideal es que permita el desplazamiento de las líneas en la pantalla *(scrolling)* y dé una longitud de línea de 80 caracteres. El software Micronet no es apropiado, ya que no permite esto. Se pueden utilizar micros con visualizaciones de menos de 80 columnas, pero son incómodos. Existen a la venta algunos paquetes excelentes; por ejemplo, Termi y Communicator son chips de ROM para el BBC Micro.

Es necesario que el modem sea capaz de comunicarse con el PSS de la British Telecom (Compañía de Telecomunicaciones Británica) y puede ser de 300/300, 1200/75 o 1200/1200 baudios. Un acoplador acústico Micronet resulta bastante satisfactorio. El PSS es un servicio de conexión en red que permite que el usuario establezca contacto con ordenadores anfitriones distantes, a menudo por el costo de una llamada telefónica local.

El procedimiento para ganar acceso al ordenador central es sencillo y directo. Con el software emulador de terminal ejecutándose, usted llama al núme-

Adrian Morgan

ro de teléfono de la central telefónica del PSS en la que está registrado, conecta su modem y digita su identidad. Luego usted entra la dirección del PSS del DEC10 de la Universidad de Essex. Allí se ha preparado una cuenta especial para permitir el acceso gratuito al MUD desde la medianoche hasta las 7 de la mañana de los días laborables, y desde las 10 de la noche a las 7 de la mañana los fines de semana, cuando la carga del ordenador es menor. Conéctese (*log*) a esta cuenta especial y luego entre RUN MUD.

El programa se actualiza continuamente, de modo que lo primero que el usuario ve es la fecha de la última versión. Se elige un nombre de personaje con el cual se jugará y se le dice al MUD qué es este personaje. Si está jugando al MUD por primera vez, entonces se creará un nuevo personaje para usted, que será clasificado como un Aprendiz. Los niveles de experiencia son:

Nivel	Puntos	Hombre	Mujer
1	0	Aprendiz	Aprendiza
2	400	Guerrero	Guerrera
3	800	Héroe	Heroína
4	1600	Campeón	Campeona
5	3200	Superhéroe	Superheroína
6	6400	Encantador	Encantadora
7	12800	Hechicero	Hechicera
8	25600	Mago	Maga
9	51200	Legendario	Legendaria
10	102400	Brujo	Bruja

Si usted acaba su sesión sin haber sido muerto, el programa anotará los puntos que haya acumulado y la próxima vez que juegue continuará a partir de esa puntuación. El usuario consigue puntos e incrementa su nivel de experiencia arrojando piezas del tesoro en el pantano, superando problemas ocasionales, destruyendo diversos seres repulsivos como ratas y zombies, y ganando una lucha contra otro jugador. Si en el curso de una pelea con otro participante usted muere, su personaje queda eliminado del juego y debe volver a comenzar como Aprendiz, sin ningún punto.

El usuario empieza el juego en un "estrecho sendero a través del campo". Al digitar "WHO" (quién) se le proporcionará una lista en pantalla de todos los que están jugando en ese momento. Usted puede decidir saludar a alguno de ellos. Por ejemplo, al digitar "Jez, hola!, soy nuevo y necesito algún consejo" le transmitiría ese mensaje al terminal del jugador llamado Jez. O bien, les podría hablar a todos los jugadores al mismo tiempo digitando: "SHOUT (gritar), OK, vosotros basuras terminales, cuidaos que aquí vengo yo", pero no es aconsejable que empiece su primer juego de esta manera.

"HELP" (ayuda) le proporcionará alguna información sobre cómo moverse y una breve explicación de muchas de las órdenes. Las opciones de movimiento se explican de la siguiente forma: "Se aceptan la mayoría de las órdenes de movimiento simples, como n, so, o, arriba, saltar, más otras que tú tendrás que descubrir!"

Las instrucciones disponibles se pueden listar digitando "COMMANDS" (instrucciones). Existe mucha información disponible, e irlas transcribiendo en papel mientras las palabras van apareciendo en la pantalla del monitor no es cosa fácil. Algunos programas de emulación terminal le permitirán a usted copiar todo lo que aparece en su pantalla en disco para poder consultarlo después. Esto también le posibilitará diseñar un mapa adecuado del terreno, que se puede actualizar debidamente después de cada juego. Éstas son las instrucciones que estuvieron disponibles un cierto día traducidas al castellano:

INSTRUCCIONES

AutoQuién, ‹segundos›	Detrás	Frenético
Breve	Bicho	Adiós
Converse	Arrojar ‹ítem›	Arrojar todo
Vaciar ‹bolso›	Salidas	Huir ‹dirección›
Seguir ‹nombre›	Coger ‹ítem›	Coger todo
Dar ‹ítem› a ‹nombre›		Ir ‹dirección›
Ayuda	Ayuda ‹nombre›	Pistas
Horas	Abrazar ‹nombre›	Info
Inventario	Conservar ‹ítem›	Matar ‹nombre›
Besar ‹nombre›	Nivel	Conectarse
‹nivel›, ‹mensaje›	Mirar alrededor	Mirar ‹bolso›
Mirar ‹dirección›	Perder ‹nombre›	Meditar
NoContraseña	Contraseña	Personaje
Pronombres	AnimarQuién	Salir
"‹mensaje›"	Rehusar ‹nombre›	
Vengarse ‹ítem›	Salvar	Marcador
Gritar, ‹mensaje›	Dormir	Hechizo
Robar ‹ítem› a ‹nombre›		
decir ‹nombre›, ‹mensaje›		Deshacerse
Prolijo	Pesar ‹ítem›	Cuándo
Quién	Escribir ‹objeto›, ‹mensaje›	

El MUD es una larga aventura basada en textos, con descripciones extensas y detalladas de los lugares. Cuando se está familiarizado con el escenario se puede digitar "BREVE" y no tendrá que leer las descripciones cada vez. Los suscriptores del Prestel ya estarán al corriente de la lentitud y la insuficiencia de los gráficos de teletexto, y si bien las aventuras basadas en gráficos son una novedad interesante, los jugadores de juegos de aventuras exclusivos siempre preferirán un juego basado en textos. Una aventura compuesta sólo de textos permite una implicación en la misma mucho más imaginativa que una basada en gráficos, del mismo modo que una emisión de radio se puede disfrutar más a veces que la televisión. Otra desventaja de las aventuras con gráficos es que cada marca de micro personal requerirá una versión diferente del juego, debido a las diferentes características de gráficos de los ordenadores personales.

Es probable que muchos de los jugadores del MUD utilicen un BBC Micro, un Apple o un Spectrum, pero otros tendrán terminales de segunda mano comprados en tiendas de ocasión y, por lo tanto, la gama de máquinas que utilizan el programa es muy amplia.

Se espera que el MUD sea comercializado pronto. Los autores del programa, Richard Bartle y Roy Trubshaw, están escribiendo una versión para un VAX Computer, que será comercializada por Century Communications.

El juego también se seguirá practicando a través de enlaces telefónicos, como el PSS y el Prestel, si bien de existir suficiente demanda también se podría llegar a ofrecer por cable. Los jugadores del MUD pagarán entonces una tarifa para unirse al juego, más un pequeño recargo por cada hora.

El amo de la mazmorra
Para obtener más detalles sobre el Multi-User Dungeon, contactar con:

Richard Bartle
Department of Computer Science
University of Essex
Colchester
Essex
Gran Bretaña

Componentes primitivos

Veamos ahora cómo la comprensión de los principios de los algoritmos puede ayudarle a mejorar su programación

Un algoritmo es una serie de instrucciones que indican cómo se puede realizar un proceso. Un algoritmo describe éste en función de otros procesos que ya han sido definidos, o bien desde el punto de vista de procesos que son tan básicos que no necesitan ser definidos. De modo que, en una receta de cocina, una instrucción podría ser "preparar una salsa bechamel", habiéndose proporcionado en algún otro lugar del libro de cocina una receta de salsa bechamel (algoritmo). Otra instrucción podría ser "ponga la mezcla al fuego hasta que hierva", donde se supone que el lector puede comprender totalmente la operación de poner algo al fuego hasta que hierva. Desde el punto de vista de la programación, los algoritmos se construyen a partir de instrucciones que utilizan algoritmos (procedimientos, rutinas, funciones) escritas en algún otro lugar del programa, o bien algoritmos ya incorporados en el lenguaje (instrucciones como PRINT y DIM, o funciones matemáticas como LOG y TAN).

En este capítulo analizaremos cómo se construyen algoritmos a partir de otros algoritmos y procesos o funciones *primitivas*. Las primitivas a disposición del programador son las instrucciones y funciones de que dispone el lenguaje. A partir de éstas, se escriben algoritmos que pueden realizar cosas pequeñas (p. ej., mover un sprite o aceptar un número como entrada). Estos algoritmos se utilizan luego para construir algoritmos más generales (actualizar la visualización del juego o controlar un sistema de menú) y estos últimos se emplean a su vez como componentes de otros mayores, hasta que todo el programa, contemplado como un único algoritmo, se escribe en términos de algoritmos de nivel inferior. Este concepto es la base de la llamada programación *estructurada* o *modular*.

Diseño de algoritmos

Un algoritmo posee una entrada y una salida. Esto equivale a decir que, como proceso, el algoritmo trabaja con unos datos iniciales para producir un resultado. Estos datos iniciales se le pasan al algoritmo desde el exterior disfrazados de "parámetros", que permanecen constantes para cualquier empleo particular del algoritmo pero que podrían cambiar para usos diferentes. Pasar parámetros es un hecho familiar hasta para el programador novato, dado que el sencillo programa:

```
procedimiento(parametro1,parametro2,etc.);
```

le pasa el parámetro "Hola mundo!" al algoritmo llamado por la instrucción PRINT. Ejemplos similares son FNA(P), TAN(P), LEFT$(P$,5) y POKE P,5, donde P, P$ y 5 son todos parámetros. Del mismo modo, los resultados de un algoritmo son devueltos como parámetros. Si el lenguaje de programación que se está utilizando posee variables locales (p. ej., el PASCAL y el C), los parámetros se pasarán con una llamada a procedimiento, como en:

```
10 PRINT "Hola mundo!"
```

Al diseñar un algoritmo, es un primer paso esencial considerar el contenido de los parámetros de entrada y salida, sus tipos (entero, de coma flotante, real, serie, etc.) y sus magnitudes y rangos.

Cuando se han definido las entradas y las salidas del algoritmo, el siguiente paso consiste en pensar qué acciones realizar sobre las entradas para obtener las salidas.

Lo más obvio, y lo que más frecuentemente se menosprecia, es tomar prestado el algoritmo de algún otro sitio. Al nivel más simple, las funciones incorporadas de un lenguaje de programación proporcionan muchos algoritmos útiles, como manipulación de series, funciones trigonométricas, entrada-salida y (posiblemente) clasificación y manipulación de matrices. Aparte de esto, puede que el algoritmo necesario ya exista en algún otro de sus programas. El código para el mismo se podría incorporar al programa nuevo (es sumamente útil que cada programador cree su propia biblioteca de algoritmos). Además, existen conjuntos de algoritmos editados que con frecuencia se pueden obtener en bibliotecas públicas.

También es interesante examinar los programas que se publican en las revistas de informática para buscar rutinas que puedan ser de utilidad. Por último, hay algoritmos que se pueden aplicar en otros campos de acción y, a pesar de que jamás estuvieron pensados para la informática, resultan sumamente útiles. La sección de contabilidad de la biblioteca local puede estar llena de libros que contienen fórmulas para calcular balances y depreciaciones. Una pequeña investigación entre estos libros podría simplificar muchísimo la escritura de un programa de cuentas y es probable que el resultado final fuera mucho más fiable. Lo mismo se puede decir de otras disciplinas: ingeniería, electrónica, matemáticas, etc.

Tanto al adaptar un algoritmo ya existente como al crear uno hay ciertos criterios que se le deben aplicar a cada una de las instrucciones que el mismo contiene. Éstos son *precisión* y *eficacia*. Precisión significa que la instrucción no debe ser ambigua en ningún sentido. Es fácil introducir ambigüedad en una primera etapa, cuando el algoritmo se está escribiendo en castellano. Palabras en castellano

como "y" y "o" son muy diferentes del AND (y) y el OR (o) de la lógica booleana. Por ejemplo, si el algoritmo está pensado para seleccionar todos los nombres de una lista que empiecen con "A" *y* todos los que empiecen con "B", se podría fácilmente escribir un código como:

```
IF PRIMERALETRA = "A" AND PRIMERALETRA
="B" THEN
```

lo cual está mal, ¡porque lo que se necesita es un OR (o) lógico! (es evidente que ninguna letra puede ser una "A" y una "B" simultáneamente).

El criterio de la eficacia es la necesidad de que el programa no contenga instrucciones imposibles. Se dice que una instrucción es eficaz cuando se puede llevar a cabo con lápiz y papel en un tiempo finito. Esto significa que instrucciones como "X igual al mayor número primo" no es eficaz (porque no existe un mayor número primo).

Consideraciones generales

Existen, asimismo, criterios para juzgar al algoritmo como un todo. Un algoritmo debe *terminar*. El que ofrecemos no termina (aun cuando sus instrucciones son precisas y eficaces), y si se codificara en un programa sería un bucle sin fin:

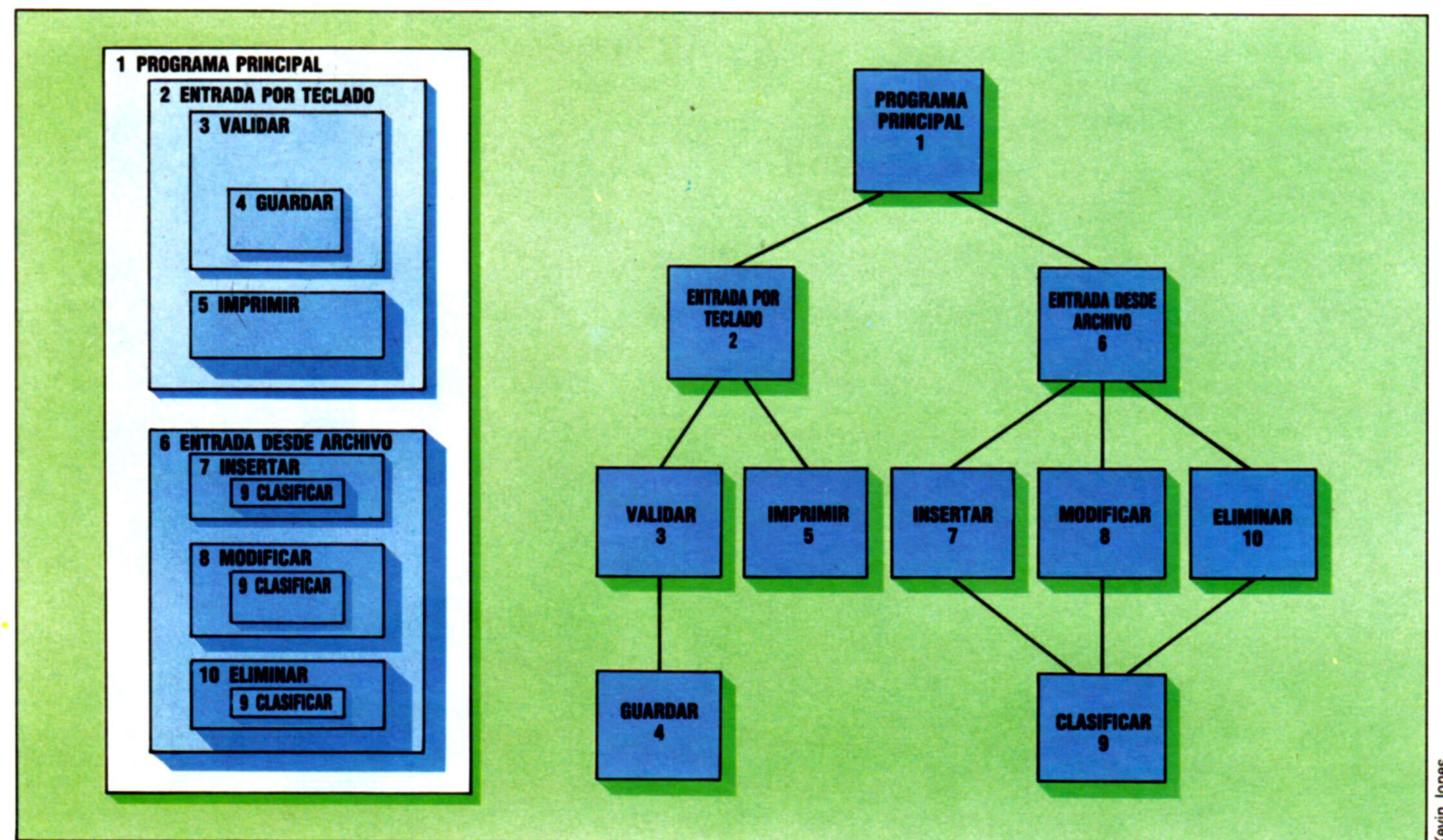
```
paso 1 poner I igual a 1
paso 2 si I > 3 entonces terminar
paso 3 ir al paso 1
```

No siempre es fácil decir si un algoritmo terminará o no pero, en general, los algoritmos que implican bucles verifican una condición determinada antes de terminar (p. ej., si I > 3, en el ejemplo dado) y es necesario verificar que sea posible satisfacer esa condición.

Eficiencia, generalidad y *elegancia* son criterios aplicados a la evaluación de algoritmos. La eficiencia se suele juzgar desde el punto de vista del tiempo de proceso y empleo de memoria. Normalmente los dos son bastante compatibles; un código rápido podría necesitar un espacio relativamente pequeño, aunque esto no es siempre así de manera obligada. Habiendo hallado un algoritmo, se lo puede "afinar" para dotarlo de mayor eficiencia modificando sus detalles. Un cálculo será notablemente más rápido y empleará menos memoria si, por ejemplo, se usa aritmética de enteros y no de coma flotante.

Generalidad es la capacidad de un algoritmo para hacer frente a muchas situaciones diferentes además de aquella para la cual fue diseñado. Vale la pena, a la larga, intentar que todos los algoritmos sean lo más generales posible. Si un programa requiere varias veces una respuesta sí/no, valdría la pena escribir una rutina que le indicara al usuario que "por favor digite S o N", aceptara la entrada, verificara si es "S" o "N", volviera a solicitarla en caso de que no fuera ninguna de ellas y que, en caso de ser correcta, retornara la respuesta entrada. Sin embargo, se podría hacer que la rutina fuera más general escribiéndola para que contemplase distintas preguntas y distintas respuestas potenciales, de modo que se la pudiera utilizar en muchas situaciones diferentes. Elegancia significa hallar algoritmos que sean simples e ingeniosos a la vez. En todos los casos es más sensato hallar algoritmos eficaces y generales que algoritmos elegantes, aunque a menudo los tres conceptos resultan muy compatibles.

Otro aspecto importante de los algoritmos es el flujo de control y de datos dentro de ellos y cómo éste se puede representar mediante diagramas de flujo. Lo analizaremos en el próximo capítulo.



Estructura y niveles de proceso
El diagrama de estructura de bloques de la izquierda ilustra claramente cómo se anidan los algoritmos de un programa, mientras que el diagrama de flujo de procedimiento, a la derecha, hace hincapié en las articulaciones y los niveles del proceso del mismo programa. Los algoritmos más "primitivos" son los que están anidados más profundamente y los que ocupan el lugar inferior en la jerarquía

Test en cascada (y 3)

Con este nuevo planteamiento, se completa el problema expuesto en el capítulo anterior

En este segundo ordinograma se ha suprimido la última pregunta de la serie eliminatoria de operaciones a realizar. Asimismo, puede comprobarse la inclusión de una comparación, tras la entrada del número, sobre si éste está entre uno y cuatro; pero aun con todo existe un segundo problema: ¿qué pasaría si se introdujera, dentro de ese rango, un número que contuviera decimales? Con la segunda pregunta (que verifica si el número es entero o no) se elimina dicha posibilidad y se tiene la certeza de que no puede entrar ningún elemento extraño. Todo dato introducido se decantará por una vía u otra. Por ejemplo, de no haberse incluido la pregunta inicial, un número erróneo, el 7, habría ido descendiendo y, al no existir control de código 4, tras el pase por el del número 3 con respuesta negativa y la decisión sobre si la segunda cantidad es 0, habría realizado, sin corresponderle, la división.

```
10 REM *********CODIGOS
20 INPUT "PRIMERA CANTIDAD";A
30 INPUT "SEGUNDA CANTIDAD";B
33 REM *********ENTRADA CODIGO
40 INPUT "CODIGO..........";C
45 REM****FILTROS
50 If C < 1 OR C > 4 THEN GOTO 40
60 IF C<> INT(C) THEN GOTO 40
70 REM *********DECISIONES
80 IF C=1 THEN PRINT"LA SUMA DA" A+B :
   END
90 IF C=2 THEN PRINT"LA RESTA DA" A-B :
   END
100 IF C=3 THEN PRINT"LA MULTIPLICACION
    DA" A*B : END
110 IF B=0 THEN GOTO 30
120 PRINT"LA DIVISION DA" A/B
125 END
```

INICIO
CANTIDAD A
CANTIDAD B
CÓDIGO C
C MENOR QUE 1 O MAYOR QUE 4 ?
SÍ
NO
C ES ENTERO ?
NO
SÍ
C ES 1 ?
SÍ
SUMA
NO
C ES 2 ?
SÍ
RESTA
NO
C ES 3 ?
SÍ
MULTIPLICA-CIÓN
NO
B IGUAL 0 ?
SÍ
NO
DIVISIÓN
FIN

Un sistema eficaz

Examinamos aquí el paquete Adam, que amplía la consola de juegos ColecoVision a un rango de ordenador personal

El sistema Adam está diseñado para apoyarse en la consola para juegos para televisor ColecoVision y los usuarios de esta unidad la puedan ampliar para satisfacer todas las especificaciones del Adam mediante unidades accesorias. De hecho, hasta la fecha todos los Adam que están a la venta se basan en el módulo ColecoVision, aunque hay planificado un Adam individual más compacto.

Cuando se desempaquetan por primera vez las cajas que contienen el sistema Adam, la cantidad de componentes resulta un poco desconcertante. El módulo ColecoVision se engancha en una bandeja plástica grande y el "módulo de memoria" (que contiene asimismo el microprocesador Z80A y la unidad de cinta) se conecta con la unidad de juegos a través de una puerta de ampliación. El módulo se mantiene en su sitio mediante unos clips que lo sujetan a la bandeja. El teclado se conecta mediante un trozo de cable en espiral y uno de los dos controladores de juegos que se proporcionan se puede insertar en una ranura a uno de los lados del teclado para actuar como un teclado numérico. La impresora está conectada al módulo de memoria, y la potencia para todo el sistema se suministra por un cable que va de la red eléctrica a la impresora.

El teclado posee 75 teclas, incluyendo seis teclas de función, un juego de teclas de control para el procesador de textos, un grupo para el cursor y las alfanuméricas habituales. Está bien diseñado y es ligero y fácil de utilizar sobre el regazo.

Cuando se conecta por primera vez el sistema, el Adam está en la modalidad "máquina de escribir electrónica". Pulsando una tecla se visualiza simultáneamente el carácter correspondiente en la pantalla y se imprime en la impresora. La pantalla es una representación de una hoja de papel y del rodillo portapapel de una máquina de escribir. La utilidad del Adam en esta modalidad es limitada, pero la pulsación de una tecla coloca al sistema en modalidad "SmartWriter" (escritor eficaz).

SmartWriter es un procesador de textos simple pero efectivo que obviamente fue diseñado teniendo en cuenta al principiante. Se mantiene la visualización "rodillo de máquina de escribir" y la línea inferior de la pantalla visualiza las instrucciones destinadas a las teclas de función. El uso adecuado de estas teclas, junto con las diversas teclas de control del procesador de textos del teclado, hace que la operatoria del SmartWriter sea sumamente sencilla; de hecho, el manual es en gran parte superfluo, ya que las opciones del menú conducen al usuario a través de todos los pasos necesarios para almacenar, visualizar e imprimir el texto. El formato de la pantalla es de 36 columnas por 20 líneas, pero se pueden manejar líneas más largas utilizando la pantalla como una "ventana" de todo el texto.

El principio de la unidad de cinta es similar al del microdrive de Sinclair, pero posee mayor capacidad (supuestamente 256 Kbytes por cinta). Es mucho más veloz que la cinta de cassette común.



Chris Stevens

La familia Adams
El Adam es una máquina de todo o nada. Se vende como un sistema completo: micro, unidad de cinta de gran velocidad, impresora de rueda margarita, teclado, dos palancas de mando y tres paquetes de software (dos juegos excelentes y un procesador de textos). Todo esto hace que el Adam sea un ordenador personal bastante potente; sin embargo, esencialmente es una forma de mejorar una unidad de juegos para televisión



Jugando con Buck
El juego Buck Rogers viene en cinta para la propia unidad del Adam y la aprovecha al máximo. El juego está dividido en varias fases, que son demasiado largas para caber en la memoria simultáneamente, de modo que el Adam carga una fase desde la cinta y luego, mientras se está jugando, carga la siguiente, y así sucesivamente. Al final de cada juego la cinta se rebobina automáticamente y si se ha conseguido una puntuación elevada, ésta se graba en la cinta. Por consiguiente, la tabla de puntuaciones refleja las mejores marcas que se han conseguido, no sólo la mejor desde que se conecta el ordenador

Asimismo, está totalmente bajo el control del ordenador, por lo que se puede rebobinar la cinta por programa para encontrar un archivo determinado. Las cintas (o los "paquetes de datos", en la jerga de Coleco) ya se suministran formateadas, de modo que no se pueden utilizar cassettes de audio normales. Cuando se enciende el Adam, se carga el programa de la cinta que en ese momento esté en la unidad, que puede ser un paquete de juegos, la cassette de SmartBASIC o un programa del usuario. Si la unidad está vacía, el sistema se coloca automáticamente en "máquina de escribir electrónica".

La impresora de rueda margarita utiliza bobinas de papel o bien papel en hojas sueltas, y su calidad de impresión es alta.

A pesar de los 80 Kbytes de RAM, el SmartBASIC, basado en cassette, sólo deja 28 Kbytes para el usuario. El SmartBASIC es similar a la versión del BASIC de Apple y es igualmente fácil de utilizar, con buenos gráficos y una selección de 16 colores. El manual de BASIC, sin embargo, es verdaderamente delirante, escrito en un tono condescendiente y lleno de expresiones infantiles, tales como *boo-boo* por "error". Pero las similitudes entre el SmartBASIC y el BASIC Apple hacen que los usuarios dispongan de una gran cantidad de libros y otro material al cual remitirse para consulta.

A primera vista el sistema Adam parece ofrecer una excelente relación calidad-precio. El software incorporado es ideal para escribir cartas, informes o artículos cortos y el sistema de cinta es la simplicidad personificada. Una ventaja adicional es la gran cantidad de software para juegos de gran calidad que hay disponible (en el Adam no sólo se pueden utilizar cartuchos ColecoVision, sino que comprando un adaptador también se podrá disfrutar de la inmensa gama de juegos de Activision y Atari). El software más serio se está vendiendo en cinta, si bien el número de paquetes disponibles es más bien limitado. Entre ellos se cuentan una hoja electróni-

Chris Stevens

Teclado separado
Sólo hay dos micros personales que se venden con teclado separado, aun cuando esto se considera esencial en el caso de los ordenadores de oficina. El teclado del Adam incluye teclas de función y teclas exclusivas para el software de tratamiento de textos que viene con el micro

Mando de la palanca de mando
Aunque el brazo del mando de la palanca es mucho más corto que el de la mayoría de ellas, es suficiente para usarlo con seguridad

Almacenamiento de palancas de mando
Aquí se pueden guardar, cuando no se las utilice, dos palancas de mando. Éstas se enchufan en conectores situados junto a este sector

Sistema de videojuegos ColecoVision
Esta unidad se vende sola como un ordenador exclusivo para juegos, pero se puede ampliar, convirtiéndola en un ordenador, mediante la adición de otras unidades

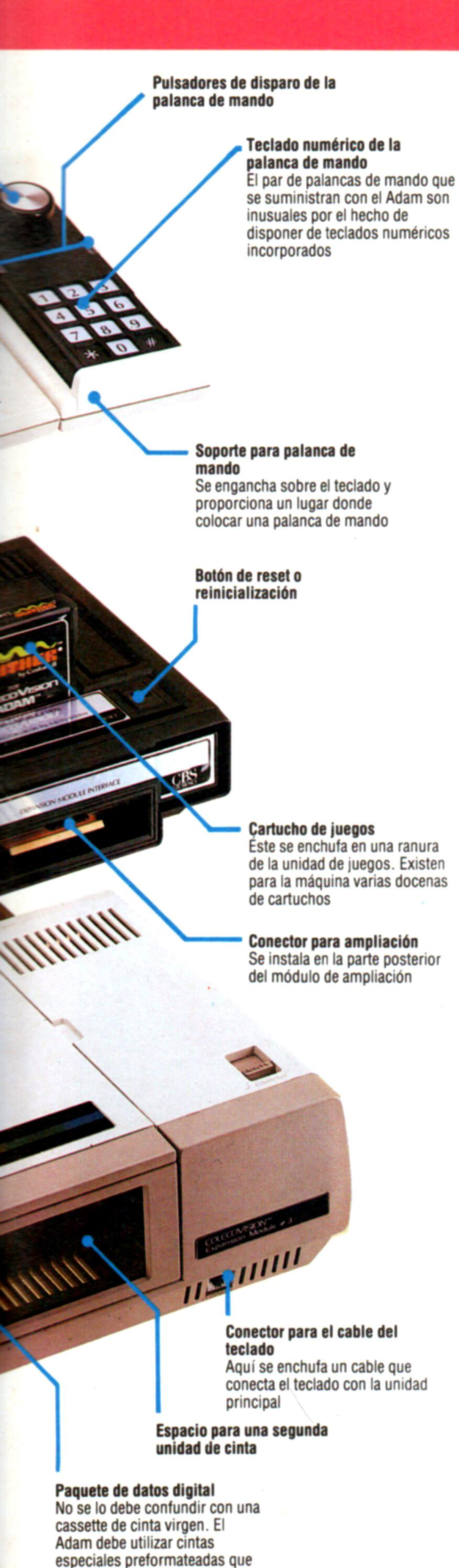
Bandeja del equipo
Se encaja en la parte inferior de las dos unidades para mantenerlas firmemente unidas

Conector para impresora
En este conector se enchufa la impresora que viene incluida con el Adam. A través de ella se le suministra corriente al Adam, de modo que la impresora siempre debe estar conectada, incluso cuando no se la utilice

Unidad de cinta digital
Podría parecer una grabadora de cassette normal, pero se trata de una unidad de cinta de gran velocidad controlada totalmente por el ordenador

Chris Stevens





Pulsadores de disparo de la palanca de mando

Teclado numérico de la palanca de mando
El par de palancas de mando que se suministran con el Adam son inusuales por el hecho de disponer de teclados numéricos incorporados

Soporte para palanca de mando
Se engancha sobre el teclado y proporciona un lugar donde colocar una palanca de mando

Botón de reset o reinicialización

Cartucho de juegos
Éste se enchufa en una ranura de la unidad de juegos. Existen para la máquina varias docenas de cartuchos

Conector para ampliación
Se instala en la parte posterior del módulo de ampliación

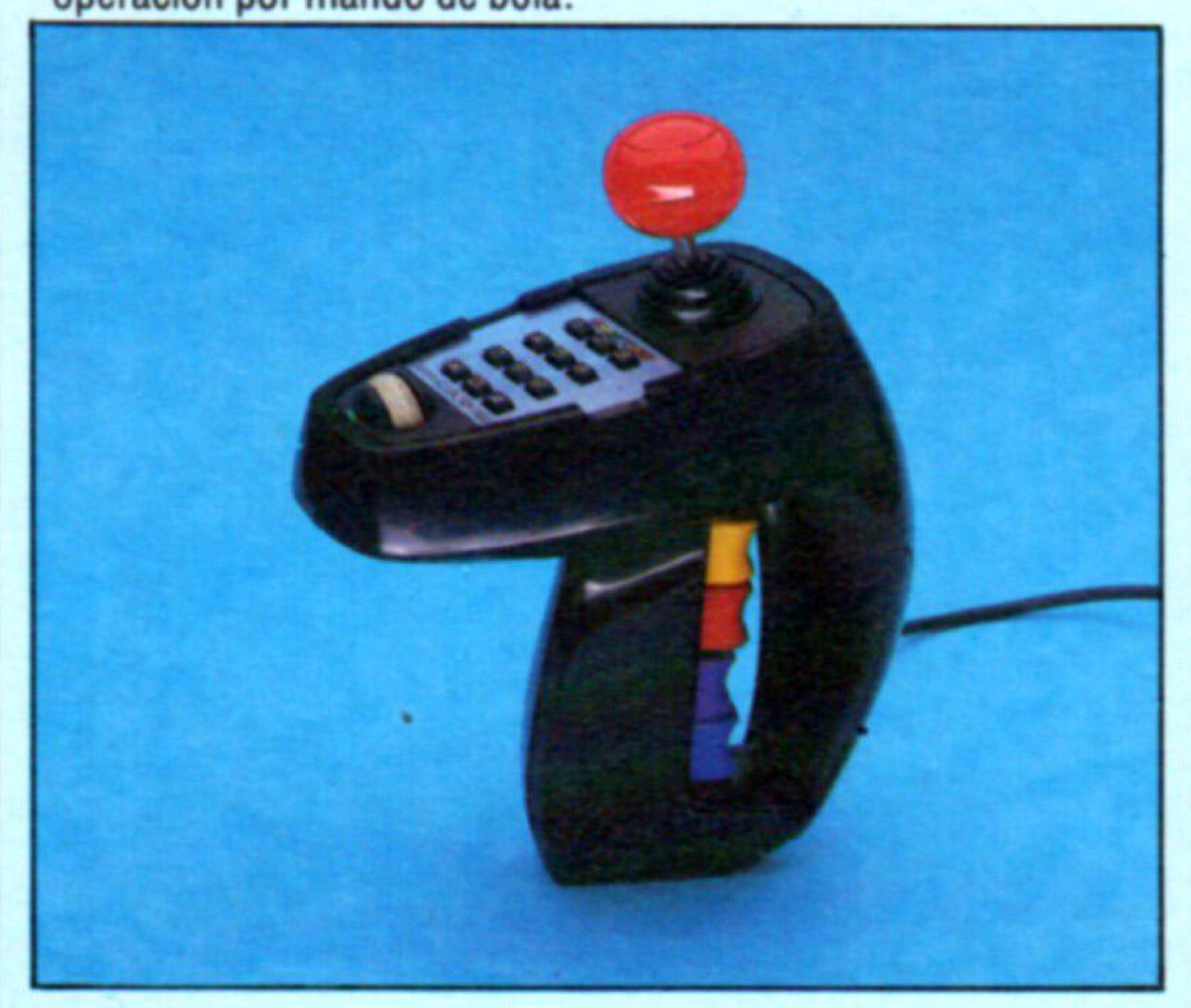
Conector para el cable del teclado
Aquí se enchufa un cable que conecta el teclado con la unidad principal

Espacio para una segunda unidad de cinta

Paquete de datos digital
No se lo debe confundir con una cassette de cinta virgen. El Adam debe utilizar cintas especiales preformateadas que almacenan 256 Kbytes cada una



Las palancas de mando del Adam
Con el Adam se suministran como estándar dos palancas de mando. Una de ellas actúa asimismo como teclado numérico y se coloca en un soporte especial a un lado del teclado. Coleco también comercializa dos sistemas *deluxe* para control de juegos. El conjunto Super Action Controller se compone de un par de palancas de mando de acción sencilla con teclados numéricos y gatillos tipo fusil en vez de los pulsadores de disparo normales. Éstas se instalan con conectores Atari de tipo D estándares y, por consiguiente, se pueden utilizar con otros sistemas. Se incluye un cartucho de juegos Baseball gratuito. El Roller Controller es un panel de control de juegos de tipo recreativo. Posee dos palancas de mando Coleco estándar y un controlador "de bola" para una respuesta rápida y un control exacto. Dos juegos de pulsadores de disparo dobles permiten la acción de dos jugadores, y un conmutador de modalidades permite seleccionar entre operación por palanca de mando u operación por mando de bola.



ca, una base de datos y una versión del lenguaje LOGO. La adición de la ampliación de memoria prometida, de 144 Kbytes, debería proporcionar algún tipo de capacidad para CP/M.

Pero también hay inconvenientes. La calidad de impresión no es todo lo buena que uno esperaría de una impresora de rueda margarita; la impresora, además, es notablemente lenta y ruidosa, hay una grave carencia de RAM para el usuario y el hecho de que el BASIC se deba cargar desde cinta pronto resulta fastidioso. Con la reciente caída de los precios de las impresoras de rueda margarita, sería posible reunir un sistema que superase al Adam en rendimiento pero costará aproximadamente lo mismo. Dicho esto, parece que el Adam es una adquisición interesante en el caso de que el usuario ya posea la consola de ColecoVision. Para quienes desean adquirir un sistema de ordenador personal con capacidades para tratamiento de textos, el Adam es, por cierto, digno de consideración..., pero bien podría haber otras alternativas mejores.

COLECO ADAM

DIMENSIONES

381 × 279 × 102 mm (unidad de ampliación de memoria)
381 × 355 × 152 mm (impresora)
381 × 152 × 51 mm (teclado)

CPU

Z80A

MEMORIA

80 Kbytes de RAM, de los cuales se utilizan 16 K para la visualización en video. Ampliable a 144 K con el paquete de RAM opcional

PANTALLA

36 columnas × 20 filas (31 columnas × 24 filas en BASIC). 256 × 159 pixels en alta resolución, con 16 colores

INTERFACES

Conector para impresora SmartWriter, enchufe para teléfono modular Adam Net, tres ranuras para ampliación, interface para ampliación ColecoVision

LENGUAJES DISPONIBLES

SmartBASIC, suministrado en cassette; sistema operativo CP/M

TECLADO

75 teclas, trazado QWERTY, teclas esculpidas tipo máquina de escribir. Incluye 6 teclas de función programables

DOCUMENTACION

Con el Adam vienen tres manuales: uno de instalación de 64 páginas, una guía para el programa de tratamiento de textos SmartWriter y un manual de programación en SmartBASIC. Por lo general la documentación es fácil de seguir, pero algo recargada e insustancial. El manual de BASIC es insólito

VENTAJAS

La gran disponibilidad de cartuchos de juegos ColecoVision y la gran calidad de los mismos hace que el Adam resulte ideal para los amantes de los juegos. El software de tratamiento de textos es sencillo de utilizar

DESVENTAJAS

Impresora lenta y ruidosa. Paquetes de datos disponibles sólo de Coleco. Software disponible en cintas limitado

Lento pero seguro

En una serie de lecciones estudiaremos la construcción de un juego utilizando el BASIC BBC. En cada capítulo añadiremos una parte del programa total

El BASIC BBC le ofrece al programador dos ventajas respecto al BASIC Microsoft estándar: es rápido en ejecución y posee características que permiten la confección de programas estructurados. La esencia de un programa estructurado consiste en desarrollar pequeñas secciones independientes de código que se puedan depurar individualmente antes de ensamblarlas en un programa más grande. Todo programa en BASIC se puede estructurar hasta cierto punto mediante el empleo de subrutinas para codificar cada módulo del programa, pero el BASIC BBC posee tipos especiales de subrutinas, llamadas *procedimientos*. A éstos se los puede considerar como bloques de código diseñados para realizar un trabajo específico dentro del programa. Por ejemplo, imaginemos una porción de programa que ha de hacer una pausa entre cada instrucción durante un tiempo dado. En BASIC estándar, ésta se podría escribir utilizando un *bucle ficticio*, es decir, un bucle que no hace nada más que consumir tiempo en su ejecución y que podríamos transcribir en la siguiente manera:

```
10 PRINT "PRIMERA SECCION"
20 FOR I=1TO100:NEXT I
30 PRINT "SEGUNDA SECCION"
40 FOR I=1TO100:NEXT I
50 PRINT "TERCERA SECCION"
60 FOR I=1TO100:NEXT I
70 PRINT "CUARTA SECCION"
80 END
```

No obstante, sería un enfoque mejor colocar el bucle de retardo dentro de una subrutina:

```
10 PRINT "PRIMERA SECCION"
20 GOSUB 100
30 PRINT "SEGUNDA SECCION"
40 GOSUB 100
50 PRINT "TERCERA SECCION"
60 GOSUB 100
70 PRINT "CUARTA SECCION"
80 END
100 REM**SUBRUTINA**
110 FOR I=1TO100:NEXT I
120 RETURN
```

Procedimiento de rutina
A diferencia de un diagrama de flujo, este diagrama de estructura refleja la estructura de procedimientos del programa en vez de su flujo de control. Una cápsula indica el comienzo de un bucle REPEAT...UNTIL; los rombos son recuadros de decisión: cuando la condición fracasa, el bucle que lo encierra continúa. Los números de Nivel reflejan la estructura de bloques del programa: todos los comienzos de bucles y las llamadas a procedimientos abren un nuevo bloque del programa y un nivel lógico inferior. Compare esto con el diagrama de la página 867

NIVEL 0: MINAS
Repetir hasta que indicador de final=1
NIVEL 1: Título; Preparación; Bucle; Final del juego
Repetir hasta que tiempo > 1200 o fin
NIVEL 2: Inicializar variables; Definir caracteres; Dibujar borde; Colocar minas; Est. tiempo; Est. hombr.; Est. marc.; Posicionar caracteres; Comprobar teclado; Actualizar tiempo; Tirotear
Tecla pulsada; Punto=1
NIVEL 3: Continuar; Desplazar; Línea; Continuar; Explotar
Punto (detec.)=2; Punto (homb.)=2
NIVEL 4: Convertir; Continuar; Mina hallada; Continuar; Explotar; Restaurar
Restaurar

Janet Barrance



Reemplazando la subrutina por un procedimiento en BASIC BBC se obtiene el siguiente código:

```
10 PRINT "PRIMERA SECCION"
20 PROCretardo
30 PRINT "SEGUNDA SECCION"
40 PROCretardo
50 PRINT "TERCERA SECCION"
60 PROCretardo
70 PRINT "CUARTA SECCION"
80 END
100 REM**DEFINIR PROCEDIMIENTO**
110 DEF PROCretardo
120 FOR I=1TO100:NEXT I
130 ENDPROC
```

Existen numerosas similitudes entre la construcción de subrutinas y la construcción de procedimientos; por ejemplo, ambos se codifican después de la sentencia END pero se los puede llamar repetidamente desde dentro del programa principal. La ventaja fundamental del procedimiento es que se lo llama por su nombre en vez de por su número de línea. La DEFinición del PROCedimiento puede comenzar en cualquier lugar después de la sentencia END.

Si deseáramos que el programa que hemos dado como ejemplo fuera capaz de esperar durante períodos diferentes antes de cada sección, entonces se podría utilizar una ventaja más importante de los procedimientos: la capacidad de pasar parámetros a una definición de procedimientos. Vamos a suponer que deseamos que la pausa entre la primera y la segunda sección sea de 100 bucles, la pausa entre la segunda y la tercera sección de 200 bucles y la pausa entre la tercera y la cuarta sección de 175 bucles. En BASIC estándar sería necesario asignarle el valor a I cada vez antes de llamar a la subrutina. Utilizando procedimientos, el valor se puede pasar mediante unos paréntesis al final de la sentencia de llamada:

```
10 PRINT "PRIMERA SECCION"
20 PROCretardo(100)
30 PRINT "SEGUNDA SECCION"
40 PROCretardo(200)
50 PRINT "TERCERA SECCION"
60 PROCretardo(175)
70 PRINT "CUARTA SECCION"
80 END
100 REM**DEFINIR PROCEDIMIENTO**
110 DEF PROCretardo(N)
120 FOR I=1TON:NEXT I
130 ENDPROC
```

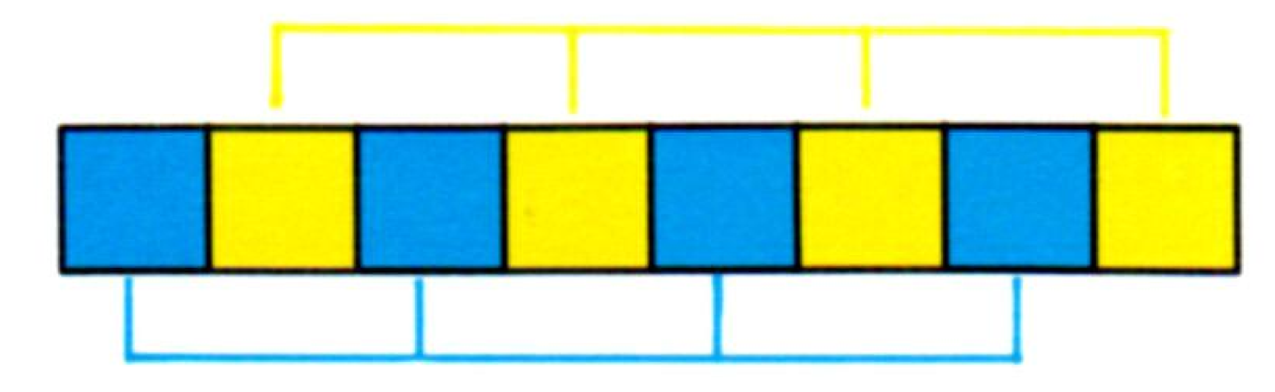
Dentro de un procedimiento se pueden pasar varios parámetros, debiendo quedar separados mediante comas. También se pueden utilizar como parámetros nombres de variables para pasar el valor de la variable en el momento en que se llama al procedimiento.

El juego que construiremos es para un jugador y utiliza las teclas de control del cursor del teclado para desplazar un detector de minas por un campo de minas. Se consiguen puntos por cada mina que se manipula con éxito. Existen, no obstante, varias cosas que dificultan su avance a través del campo de minas. La preocupación fundamental es su ayudante, que representa todos los movimientos suyos. Mientras usted va por ahí ocupándose de las minas, debe asegurarse de que él no pise una. A usted también le está disparando un francotirador y existe un límite de tiempo de dos minutos para el juego. En la versión final, tendrá cuatro ayudantes voluntarios por juego y una opción de factor de dificultad de 0, el más fácil, a 9, el más difícil.

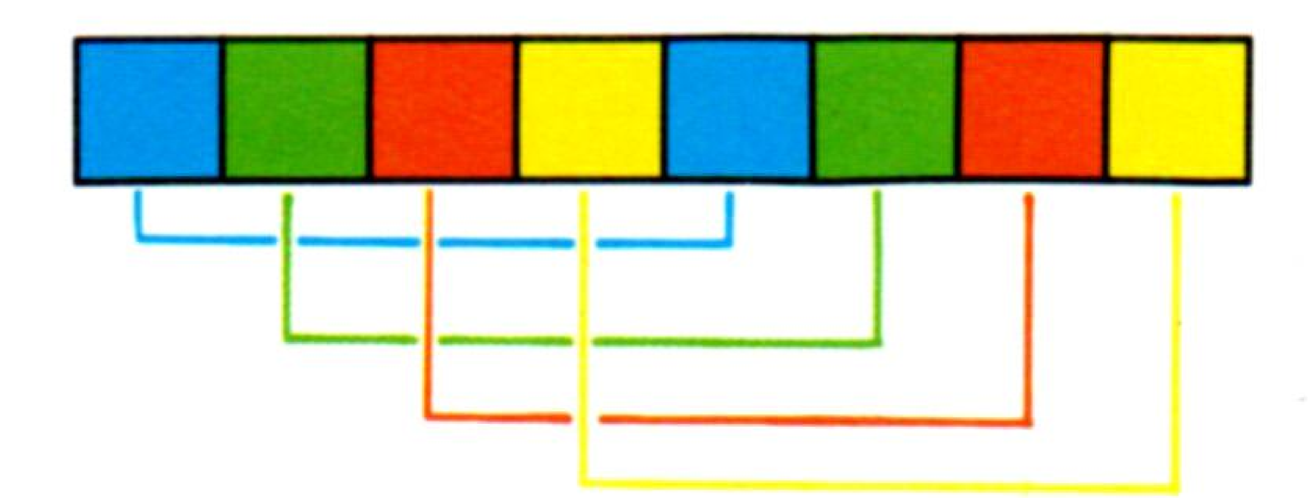
Dado que el BASIC BBC emplea procedimientos, los diagramas de flujo no son de gran utilidad. En cambio, se puede utilizar un *diagrama de estructura* para ilustrar los procedimientos que se requieren y cómo se acoplan entre sí para conformar el programa final. En nuestro diagrama los bucles REPEAT ...UNTIL se indican con forma de "salchicha". Los recuadros de decisión son los recuadros más normales, en forma de rombo, pero con los extremos superior e inferior recortados para ahorrar espacio. Se debe hacer hincapié en el hecho de que el diagrama no se trazó en su totalidad antes de que se comenzara la programación, sino que se desarrolló a partir de una serie de refinamientos sucesivos.

Antes de que podamos comenzar a definir rutinas para colocar objetos en la pantalla, debemos decidir qué modalidad de visualización en pantalla vamos a utilizar. Son tres los factores fundamentales a tener en cuenta: resolución, color y memoria. En términos generales, cuanta más información deba retener la pantalla, más memoria exigirá. De modo que resoluciones más altas y mayor número de colores significan más memoria. Si el programa es corto, esto podría no tener ninguna importancia, pero el programa que estamos diseñando es bastante extenso. También son necesarios algunos colores diferentes para distinguir las minas y el detector-ayudante y para conseguir que el juego sea visualmente atractivo. En el Modelo B se nos ofrecen dos modalidades de resolución media. La modalidad 2 nos proporciona 16 colores con los cuales jugar, mientras que la 5 sólo nos ofrece 4. Analizando cómo el BBC interpreta los patrones de bits en cada modalidad podemos ver por qué la 5 utiliza sustancialmente menos memoria que la 2.

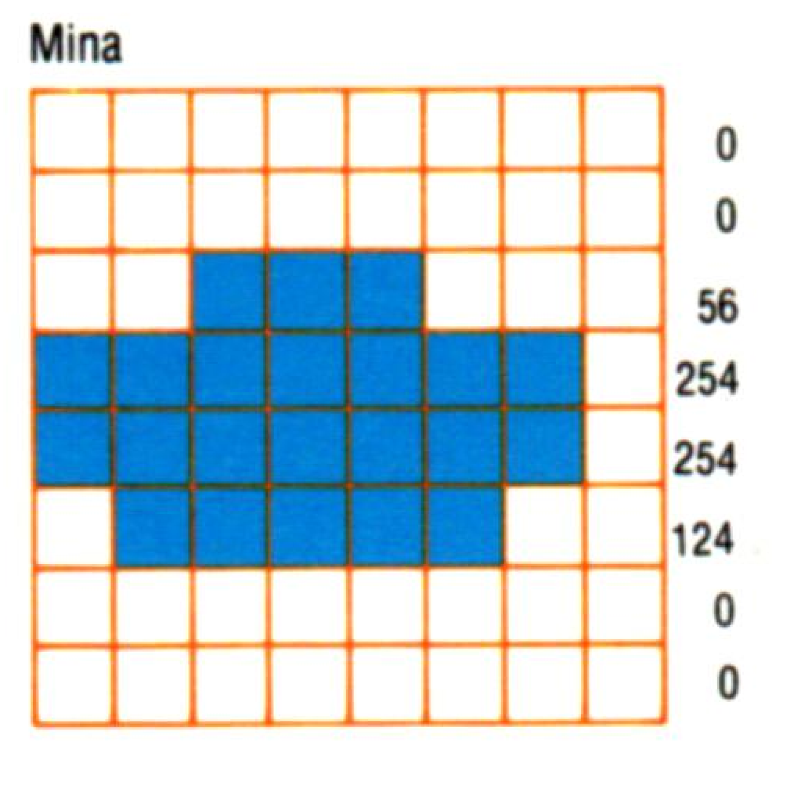
A diferencia de algunos micros, el BBC retiene la información de color y pixel encendido o apagado en un byte para cada pequeña zona de la pantalla. En la modalidad 2 se requieren cuatro bits para representar los 16 colores posibles. Por consiguiente, un byte sólo puede retener información acerca del color de dos pixels. Los bits del byte se disponen de la siguiente manera:

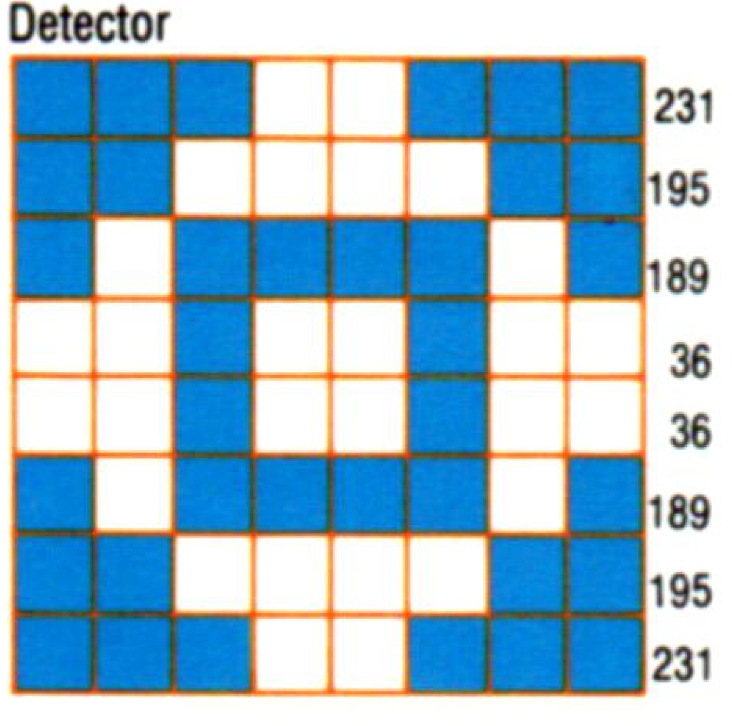


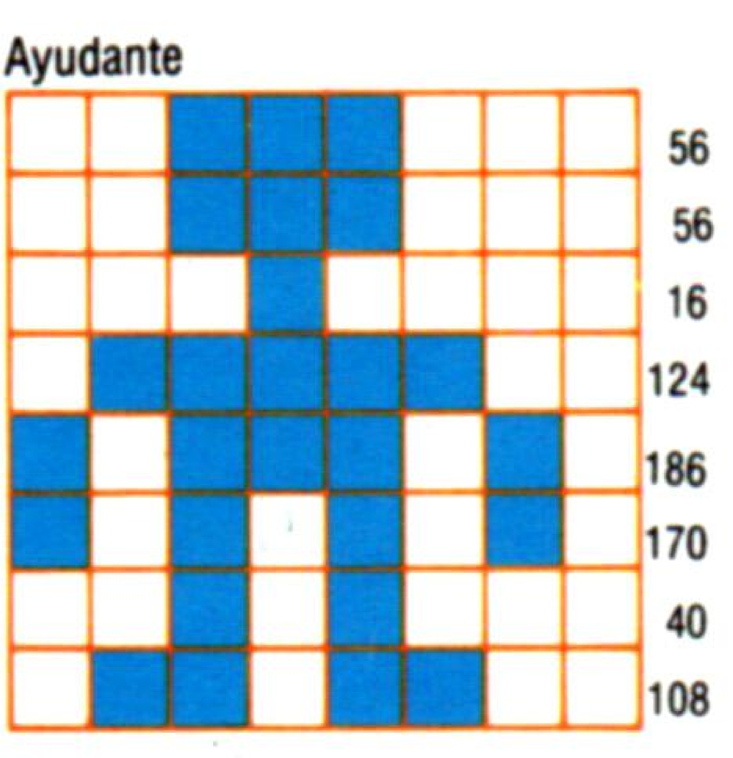
Como la modalidad 5 está limitada a cuatro colores, sólo se necesitan dos bits para retener la información de color. Así, un byte puede representar cuatro pixels, como vemos en la ilustración:



Ambas modalidades poseen una resolución de 160 por 256 pixels, por lo que el número de bytes requeridos para la memoria de pantalla de la modali-







Puntos de imágenes
Se describen los caracteres definidos por el usuario que representan la mina, el detector y el ayudante. La información de los pixels está dividida en ocho bytes, siendo cada byte la "imagen" binaria de una fila del dibujo

dad 2 es (160×256)/2=20 Kbytes, mientras que la modalidad 5 necesita (160×256)/4=10 Kbytes de memoria. Eligiendo la modalidad 5 nos concedemos a nosotros mismos 10 Kbytes extras de memoria para nuestro programa. Una ventaja adicional es que las máquinas Modelo A no disponen de la modalidad 2, ¡pero sí de la modalidad 5!

En el BBC la construcción de caracteres definidos por el usuario es muy simple. Cada carácter se compone de ocho números que representan los equivalentes decimales de cada fila. Vemos aquí los diseños de caracteres para las minas, el detector y el ayudante.

La instrucción VDU23 permite que el programador defina caracteres con códigos ASCII del 226 al 255. El segundo número de la instrucción VDU indica el código que se desea asignar a la forma definida por los ocho últimos números. Por ejemplo:

```
VDU23,224,0,0,56,254,254,124,0,0
```

define CHR$(224) como la forma de la mina. Para imprimir este carácter generamos la instrucción PRINT CHR$(224). El siguiente procedimiento define los tres caracteres a utilizar en el juego:

```
2380 DEF PROCdefinir-caracteres
2390 REM**MINAS**
2400 VDU23,224,0,0,56,254,254,124,0,0
2410 REM**DETECTOR DE MINAS**
2420 VDU23,225,231,195,189,36,36,189,195,231
2430 REM**AYUDANTE**
2440 VDU23,226,56,56,16,124,186,170,40,108
2450 ENDPROC
```

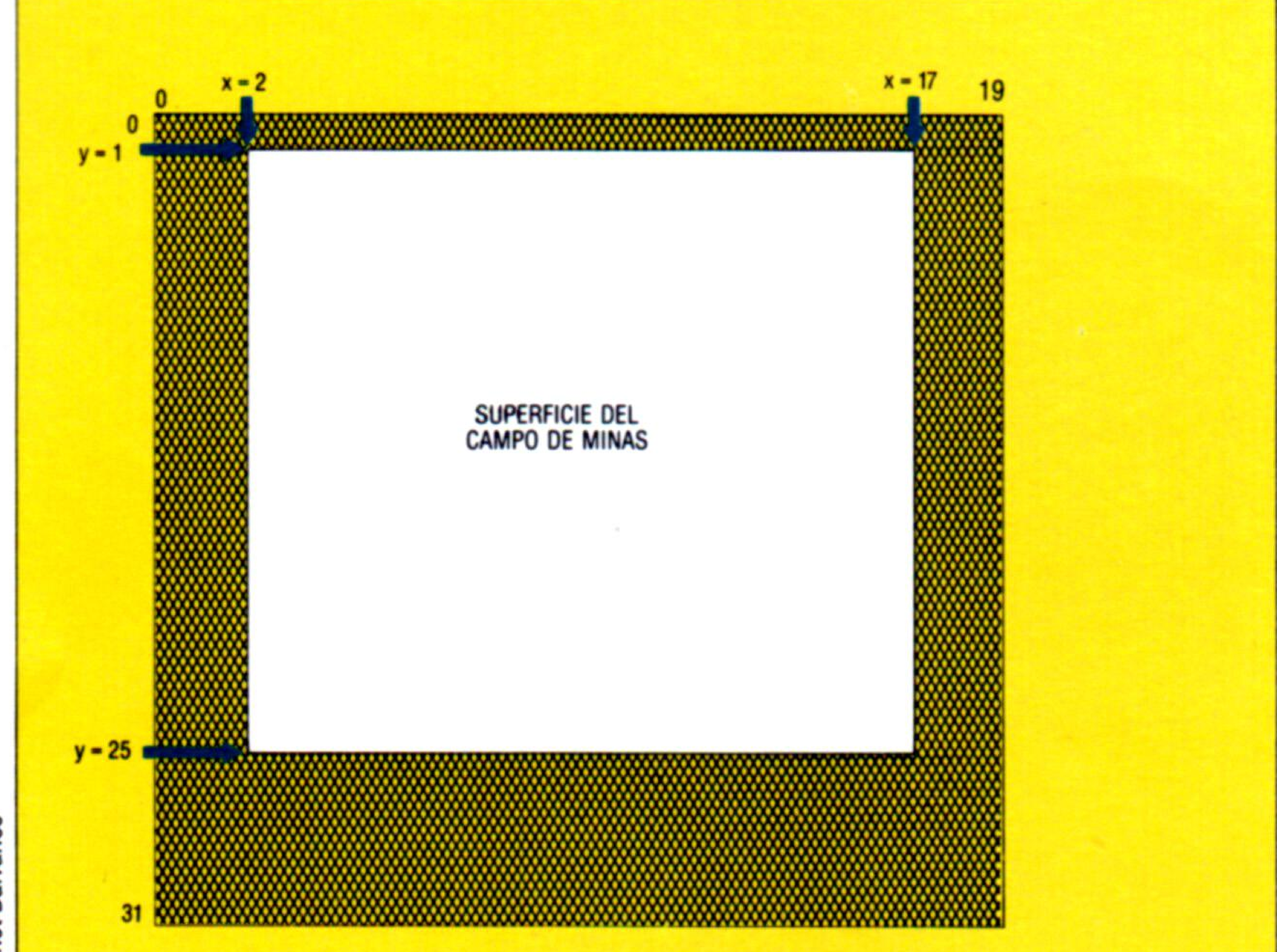
Trazado de la pantalla

Después de haber definido las formas, las podemos imprimir en la pantalla. El modo más fácil de hacerlo consiste en utilizar la instrucción PRINT TAB(X,Y). En la modalidad 5 hay 32 filas, cada una de ellas de 20 caracteres. Esto significa que X va de 0 a 19 e Y de 0 a 31. El campo de minas ha de ocupar la superficie que se indica en el diagrama.

Para disponer las minas al azar en la zona dada podemos valernos de la instrucción RND(N). Si N es un número entero, entonces RND(N) devuelve un número entero entre 1 y N. Se debe elegir la coordenada horizontal de cada mina de modo que caiga entre 2 y 17. RND(16) proporciona números del 1 al 16, de modo que RND(16)+1 seleccionará coordenadas dentro de la superficie del campo. Este procedimiento colocará tantas minas como se especifique en el valor que se le pasa como argumento:

```
2560 DEF PROCcolocar-minas(numero-minas)
2570 REM **CAMBIAR COLOR 2 A VERDE**
2580 VDU19,2,2,0,0,0
2590 FOR I=1 TO numero-minas
2600 PRINTTAB(RND(16)+1,RND(25));CHR$(224)
2610 NEXT I
2620 ENDPROC
```

Zona de peligro
El campo de minas ocupa 20 filas de 16 caracteres, y las minas se "siembran" al azar durante el procedimiento de preparación



Janet Barrance

En la modalidad 5 estamos limitados a cuatro colores y normalmente éstos son negro, rojo, amarillo y blanco, que corresponden a los números de color 0, 1, 2 y 3. Sin embargo, no tenemos que utilizar necesariamente estos colores y los podemos cambiar utilizando la instrucción VDU19. Los números del 0 al 3 se conocen como los colores de fondo *lógicos*. Cada uno de los 16 colores del BBC posee un número que no guarda ninguna relación con la modalidad que se esté empleando. Éstos se denominan números de color *reales* y en la guía para el usuario aparece una tabla de los mismos. A cualquiera de los cuatro colores lógicos se le puede asignar uno de los 16 colores reales. Para nuestro juego queremos que las minas sean verdes (número de color real 2). No deseamos el amarillo, que suele ofrecerse como el color lógico 2. La instrucción VDU19 hace esto.

El detector y el ayudante ocupan sus posiciones iniciales en las esquinas inferior izquierda y superior derecha, respectivamente. Como probablemente desearemos volver a posicionar después al detector y al ayudante, el procedimiento para posicionarlos utilizará variables (xdet, ydet) para las coordenadas del detector y para las coordenadas (xhom, yhom) del ayudante.

```
2830 DEF PROCsituar-sujetos
2840 COLOUR 1
2850 PRINTTAB(xdet,ydet);CHR$(225)
2860 PRINTTAB(xhom,yhom);CHR$(226)
2870 COLOUR 2
2880 ENDPROC
```

Las instrucciones COLOUR al principio y al final del procedimiento seleccionan el color lógico que tendrá el futuro texto. COLOUR 1 selecciona el color lógico 1 (rojo) para imprimir el detector y el ayudante, y COLOUR 2 restaura el color verde para la futura impresión. Sin embargo, antes de que se pueda aplicar este procedimiento, se deben asignar valores a xdet, ydet, xhom e yhom. Ello se realiza en otro procedimiento, junto con la inicialización de algunas variables que se usarán en otros sitios.

```
2320 DEF PROCinicializar-variables
2330 xdet=2:ydet=25:xhom=17:yhom=1
2340 xcomienzo=120:xfinal=1144
2350 cero$="000000"
2360 ENDPROC
```

Ahora todos estos procedimientos se pueden controlar mediante un breve programa de llamada:

```
5 MODE 5
10 COLOUR 2
20 PROCinicializar-variables
30 PROCdefinir-caracteres
40 PROCcolocar-minas(40)
50 PROCsituar-sujetos
60 END
```

En el próximo capítulo nos ocuparemos de sincronizar y controlar el movimiento desde el teclado.

Un piloto temerario

"Jet Pac" ha establecido un nuevo estándar de calidad en los gráficos del Spectrum, convirtiéndose en un best-seller

Los juegos recreativos en los que se trata de "aniquilar al extraterrestre" con frecuencia se ven notablemente limitados por su enfoque simplista e infantil. El diseño de un ejemplo de éxito requiere considerable destreza en cuanto a programación. *Jet Pac* es un buen ejemplo de ello.

Como suele suceder con tanta frecuencia, el escenario dibujado en la cartulina de la cassette parece más complejo de lo que es en realidad. Como piloto jefe de pruebas de la Empresa de Transportes Interestelar Acme, su tarea consiste en viajar a través de la galaxia ensamblando naves espaciales en planetas seleccionados, mientras recoge todo el oro y las piedras preciosas que puedan caer en sus manos. El ensamblaje de los cohetes se ve facilitado en gran medida por un Hydrovac Jet Pac, capaz de levantar casi todo, y que le permite manipular componentes con suma facilidad. Usted también está convenientemente armado con *quad photon laser phasers*, que se utilizan para destruir a todo extraterrestre que pueda ser tan obcecado como para quejarse de que se saquee su planeta.

La descripción del juego puede que evoque visiones de un viaje en glorioso Technicolor a través de las variadas ecologías de distintos planetas; pero, como siempre, la verdad es más prosaica. Los planetas que se visitan son virtualmente idénticos, tal es así que el héroe-piloto de pruebas probablemente experimentará un fuerte sentido de *déjà vu* a los pocos aterrizajes. Pero los extraterrestres compensan esto. Cada planeta está habitado sólo por una única especie; de hecho, habría muy poco sitio para cualquier otra especie, ya que da la impresión de que todos los extraterrestres se reproducen como conejos. Estas especies varían en su forma, desde platillos voladores hasta balones que rebotan, pero todas ellas poseen una cosa en común: el contacto físico con cualquiera de ellas significa la muerte.

Desperdigados por la superficie de cada planeta hay tres componentes de una nave espacial; éstos se han de ensamblar para que usted pueda dirigirse hacia el siguiente puerto de llamada. Para esta tarea no se necesitan destornilladores ni llaves inglesas: el usuario simplemente arroja los componentes en la base del cohete y ve cómo ante sus propios ojos la nave espacial se arma sola.

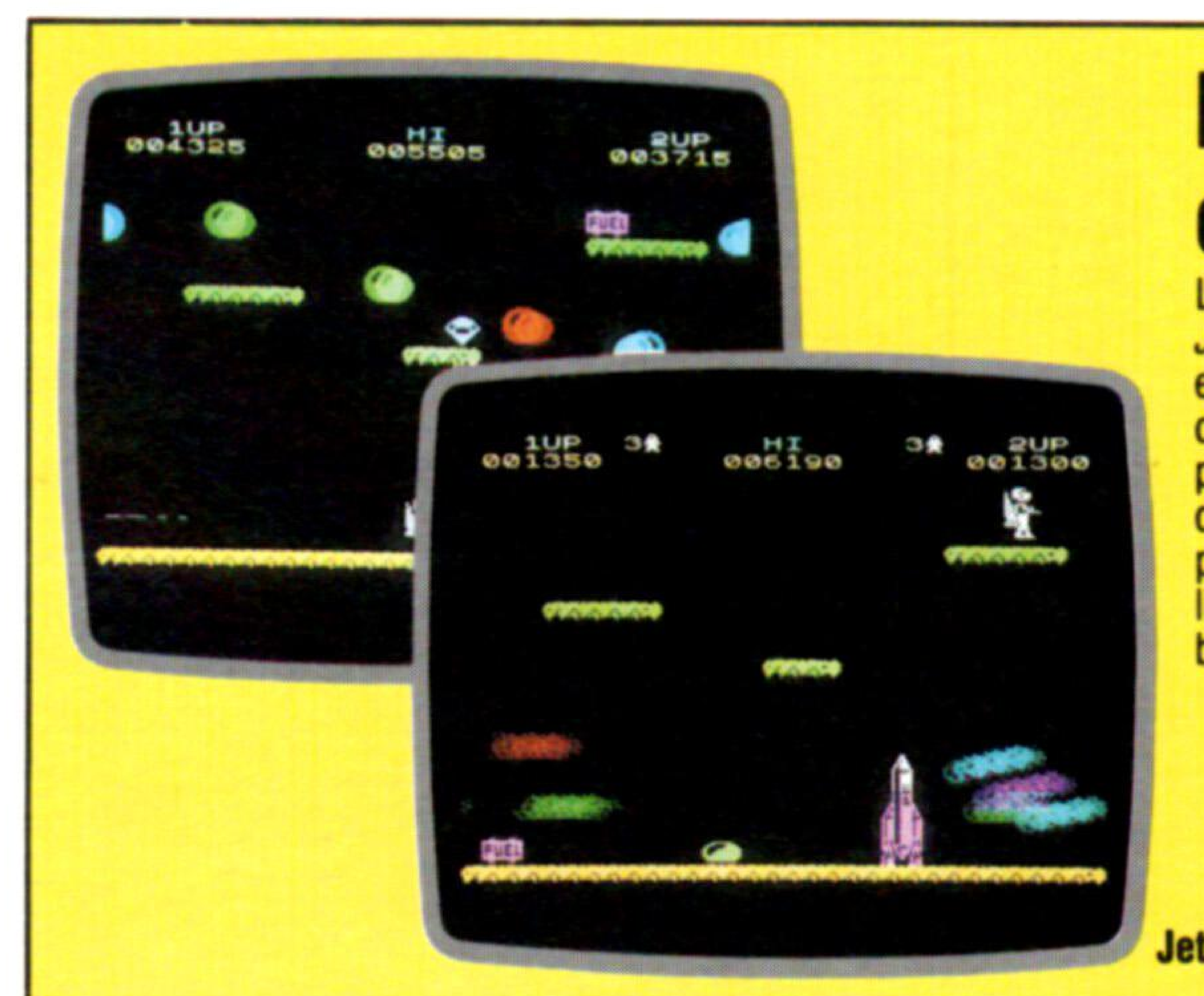
Lo que hace que este juego sea tan divertido son los extraterrestres. Su número es elevadísimo y al principio da la impresión de que avanzan amenazadora e infaliblemente hacia usted; al cabo de unos instantes comprende que, en realidad, están siguiendo caminos preestablecidos que parten desde puntos de partida aleatorios. La primera oleada de atacantes desciende lentamente, dándole tiempo para aniquilarlos con su láser letal. La segunda oleada es de pelotas rebotadoras, que rebotan a través de la pantalla entre los salientes de las rocas y el suelo. Esta mezcla de caminos preestablecidos y movimientos al azar proporciona la combinación precisa, exigiendo destreza y reflejos rápidos.

Esta clase de juego a menudo se suele arruinar a causa de una selección inadecuada de teclas de control. Aquí éstas se han elegido con notable sensatez. Para el movimiento hacia derecha e izquierda, siendo útiles tanto para jugadores diestros como zurdos, se utilizan dos teclas de la fila inferior del teclado. Cualquiera de las teclas de la segunda fila sirve para disparar los láseres, que se pueden dejar disparando de forma permanente. La fila de encima activa los motores de reacción que hacen que se ascienda, y la fila superior permite que usted flote en el aire. Un original detalle es que si no se pulsa ninguna tecla el piloto irá cayendo sobre la superficie del planeta por la fuerza de gravedad.

Los gráficos son excelentes. El dibujo del piloto de pruebas es maravilloso, al igual que el Hydrovac Jet Pac, que despide convincentes volutas de humo cuando se activa "avanzar" o "flotar". El láser atraviesa el cielo con líneas multicolores y los extraterrestres, cuando son alcanzados, explotan convirtiéndose en más nubes de humo.

El juego es bastante similar tanto en el Spectrum como en el Vic-20, si bien el formato de pantalla de este último tiene el efecto de que el piloto de pruebas resulta más bien voluminoso. En ambas máquinas, *Jet Pac* es un juego sumamente atractivo.

Jet Pac: Para el Spectrum de 16 o 48 K y el Vic-20 de 8 K ampliado
Editado por: Ashby Computers and Graphics Ltd.
Autores: Ultimate, Play The Game
Palancas de mando: Kempston Competition-Pro (Spectrum); "la mayoría de las compatibles con Commodore" (Vic-20)
Formato: Cassette



Ensamblaje espacial

La tarea del piloto de pruebas de *Jet Pac*, que consiste en ensamblar naves espaciales en diversos planetas, resulta peligrosa debido a la presencia de los decididos extraterrestres, pero el oro y las joyas que llueven del cielo constituyen una buena compensación

Jet Pac en el Spectrum

Liz Heaney

El cuadro y el marco

Tratamos aquí de un programa en lenguaje máquina para crear "ventanas" o marcos en el Spectrum donde son posibles los desplazamientos visuales



Ventana abierta
La palabra de salutación en inglés "HELLO" es desplazada o bien horizontalmente o bien verticalmente dentro de la ventana de pantalla. Siempre un pixel cada vez

Este programa en lenguaje máquina permite definir "ventanas" rectangulares sobre una pantalla gestionada por el Spectrum, así como desplazarlas (*scrolling*) a derecha e izquierda y arriba o abajo. Tales "ventanas" pueden tener sobre la pantalla toda clase de tamaños y estar situadas en cualquier lugar; no tienen por qué reducirse a cuadrados de caracteres de 8×8 pixels.

Las tablas que emplea nuestro programa en lenguaje máquina comienzan en la dirección $B004 (en decimal,45060) y se destinan al tratamiento de los parámetros de las referidas ventanas y al almacenamiento temporal de datos. Las direcciones $B000 y $B001 (en decimal, 45056 y 45057) contienen la dirección de la tabla de una ventana determinada. Cada tabla de ventana comprende 11 bytes, por lo que si necesitamos más de una ventana, la tabla de la segunda comenzará en $B00F (en decimal, 45071), la de la tercera en la dirección $B01A (en decimal, 45082), etc.

El programa de demostración en BASIC sólo emplea una ventana. Los detalles de ésta se colocan (POKE) dentro de la memoria en las líneas 180 a 230, y la rutina de inicialización de la ventana es llamada en la línea 240. Cualquier otra ventana adicional debe ser definida de este modo antes de poderse emplear. El cambio de ventanas se realiza colocando (POKE) en la memoria la dirección de la tabla de la nueva ventana, concretamente en las posiciones WT y WT+1. Las direcciones de los desplazamientos son dadas mediante la colocación (POKE) de los valores en la posición WNDWTB+DIR (*window table+direction*: tabla de ventana+dirección). Recuerde: para el desplazamiento hacia la izquierda utilice POKE 0; 1 para un desplazamiento hacia la derecha, 2 arriba y 3 abajo.

El programa en assembly comienza con la definición de constantes. PIXADR (*pixel address*: dirección del pixel) es una subrutina de la ROM del Spectrum que calcula la dirección del byte de pantalla, y el número del bit dentro de ese byte correspondiente a un punto de la pantalla definido por sus coordenadas de PLOT. La rutina toma la coordenada *y* que encuentra en el registro B y la abscisa *x* en el registro C para devolver después la dirección de la pantalla que está en el registro doble HL y la posición del bit en el registro A.

La rutina INITW lo primero que hace es comprobar que las coordenadas correspondientes a la esquina inferior derecha de la ventana se hallen por debajo y a la derecha de las coordenadas de la esquina superior izquierda. Igualmente comprueba que los bordes o márgenes izquierdo y derecho no estén en el mismo byte de la memoria de pantalla. Esto asegurará que el ancho de la ventana sea de un cuadrado de carácter, pues se precisaría código adicional para una ventana más estrecha.

Los errores de inicialización de la ventana se visualizan gracias a la rutina de mensajes de error contenida en la ROM. La instrucción RST 8 (línea 2110 del listado en assembly) llama a la rutina de la ROM y vuelve a la modalidad de instrucciones en BASIC, mientras que DEFB 25 de la línea 2120 da el mensaje "Error en parámetro Q".

La última parte de INITW se destina a calcular LFTMSK y RTMASK (*left, right*: izquierda, derecha), empleadas cuando se desplaza el byte de pantalla en los bordes de la ventana, donde parte de ese byte cae dentro de la ventana y parte fuera. Los bits individuales de las máscaras que correspondan a bits de pantalla fuera de los márgenes de la ventana se ponen a 1, y los de dentro a 0.

El programa de desplazamiento propiamente comienza donde está la etiqueta SCROLL. Lo que hace es comprobar la dirección del desplazamiento y llamar o bien a HORIZ para desplazamientos horizontales, o bien a VERT para los verticales.

Los desplazamientos a la izquierda y a la derecha operan de modo similar, por lo que en vez de escribir dos rutinas las hemos mezclado en una sola. El código adecuado a cada dirección del desplazamiento se deduce del bit 0 del byte de dirección. Para darnos cuenta del modo como funciona HORIZ vamos a detenernos en el desplazamiento hacia la izquierda.

Los dos desplazamientos, a izquierda y a derecha, comienzan por la fila superior de pixels de la ventana y de allí van bajando; por tanto HORIZ primero copia la ordenada *y* de la fila superior en una posición de memoria temporal que contiene la fila en curso. Si el desplazamiento es a la izquierda, debemos comenzar por el extremo derecho de cada fila de pixels de la ventana e ir hacia la izquierda. Para preparar esto se copian RMASK y LMASK a MASK1 y MASK2 respectivamente, se calcula y se almacena en el registro doble DE la dirección del byte de pantalla perteneciente al extremo izquierdo de la fila de pixels en curso, y por último se calcula y se almacena en el registro doble HL la dirección del byte de pantalla correspondiente al extremo derecho de la fila de pixels en cuestión. Entonces es cuando se llama a la subrutina HLNSCR para desplazar la fila de pixels. Comprueba si se ha alcanzado la fila inferior y última de la ventana, y en caso contrario toma la fila siguiente de pixels al tiempo que salta a HORIZ3 para seguir desplazando.

La subrutina HLNSCR comienza por un extremo de la fila de pixels con un byte que puede tener parte de sus bits dentro de la ventana y parte fuera de ella. Sigue con los bytes que caen por completo dentro de la ventana, hasta llegar al otro borde de ésta, donde encontrará de nuevo un byte cuyos bits puede que caigan parte fuera y parte dentro de ella. Creemos que se puede entender mejor todo esto en

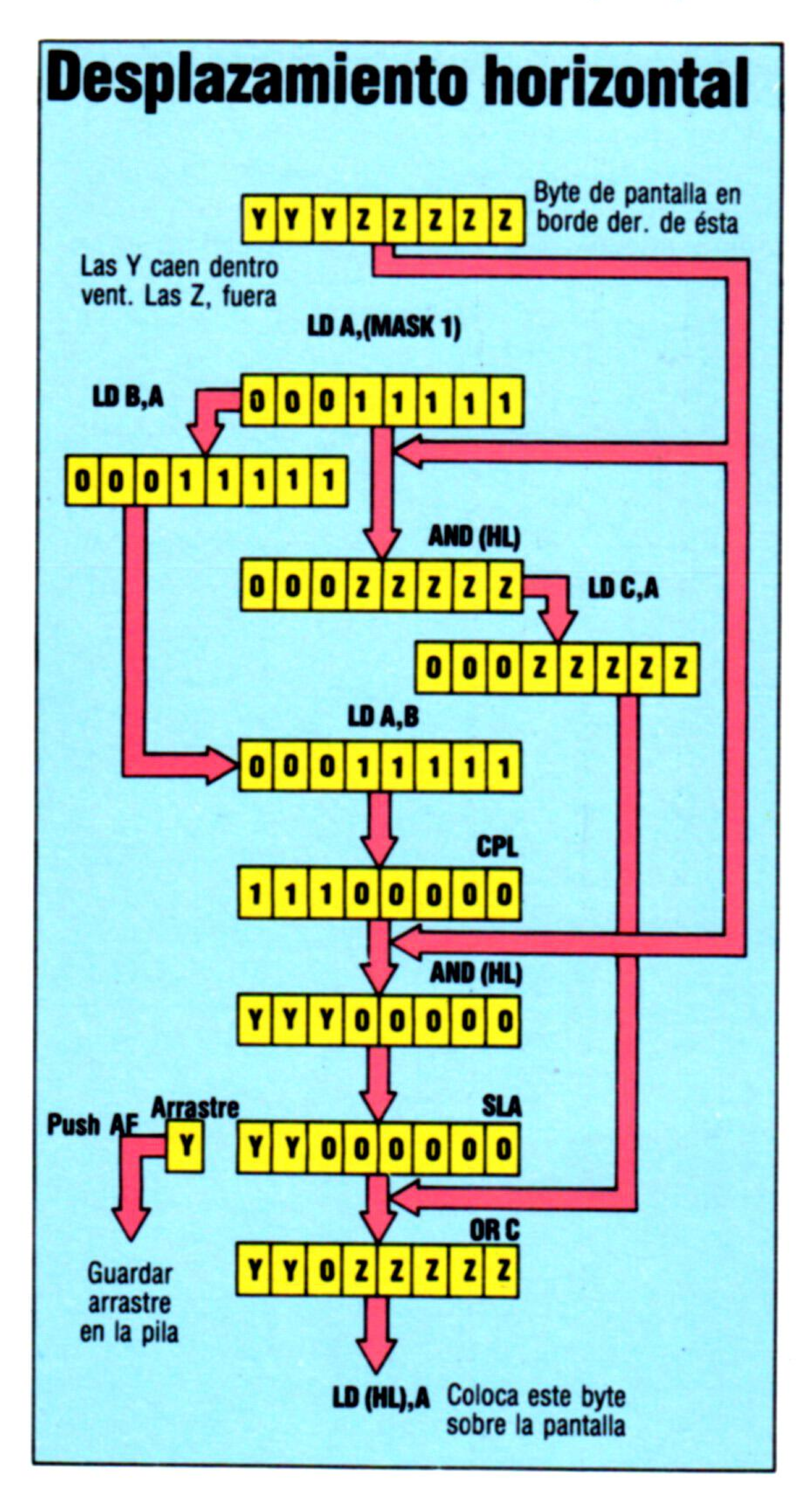


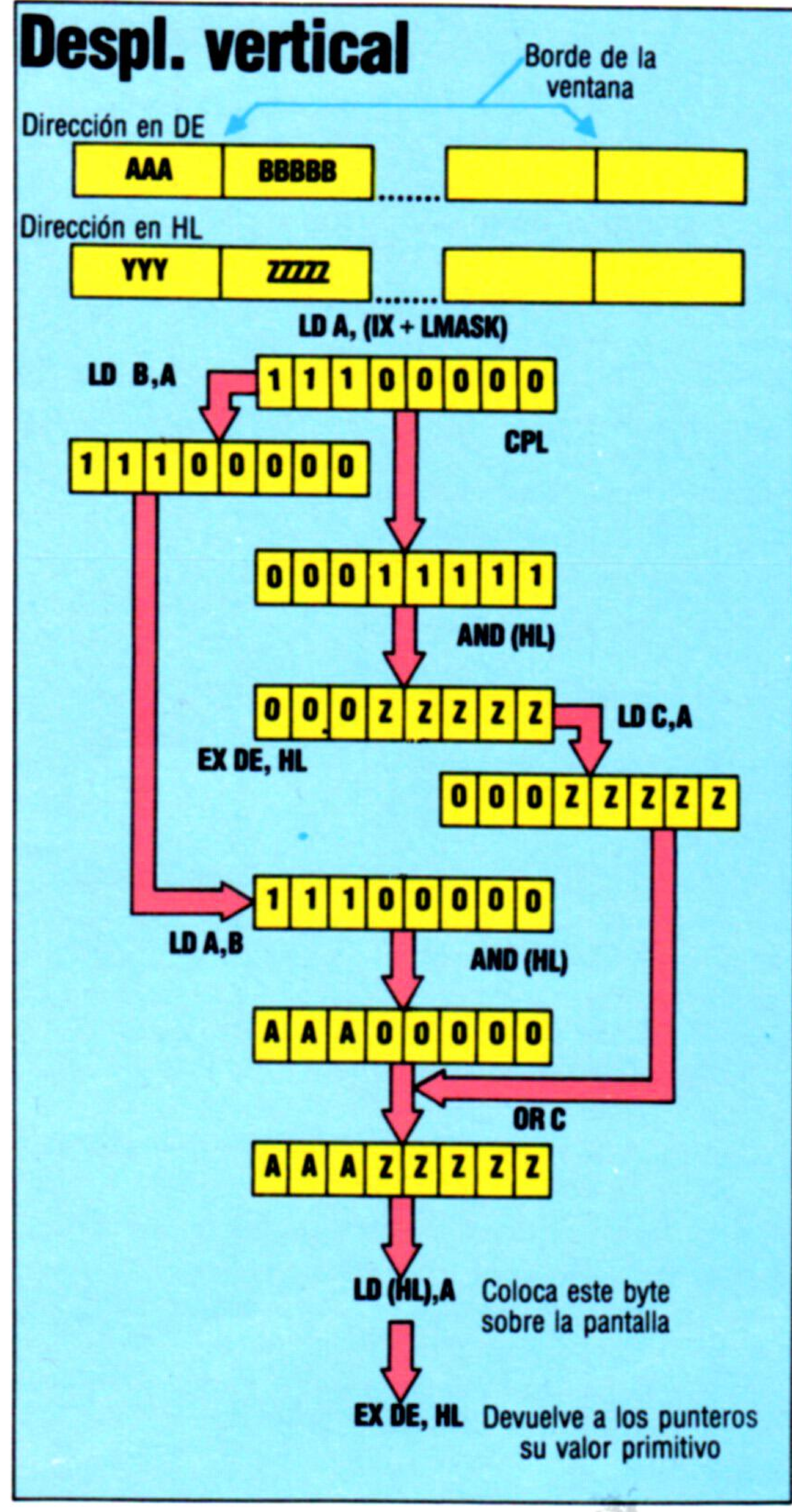
la ilustración adjunta. El fragmento de HLNSCR que comienza en NEXT desplaza los bytes que están dentro del área de la ventana. El bit descartado del byte anterior y colocado en el indicador o flag de arrastre se guardó en la pila mediante PUSH AF para devolverlo a dicho flag con POP AF. Para desplazamientos a la izquierda la rutina escoge la instrucción RL (HL) con la que el byte en pantalla se desplaza a la izquierda, quedando el bit de arrastre ahora en el último lugar de la izquierda del byte y el bit del extremo derecho de éste ocupando el flag de arrastre. Como PUSH AF conserva el flag de arrastre, dicho bit se puede llevar al byte siguiente. La rutina comprueba el final de la fila comparando los registros L y E, pues sabemos que el byte *hi* de una dirección de pantalla es el mismo para todos los bytes de la misma fila.

Las rutinas verticales se combinan de igual manera. Si examinamos lo que sucede cuando VERT está provocando un desplazamiento vertical hacia arriba, veremos que la rutina primero almacena la coordenada *y* de la fila superior en el lugar reservado a la fila en curso. Calcula después las direcciones de pantalla de los que en dicha fila corresponden a los bordes izquierdo y derecho de la ventana y determina la longitud de la fila. La rutina coloca ahora la dirección del byte del borde izquierdo en DE, y la dirección del byte que corresponde a aquél en la fila inmediatamente inferior la pone en HL antes de llamar a la subrutina VLNSCR para que realice el desplazamiento. Después VERT comprueba si ha alcanzado la parte inferior de la ventana. Si no es así, baja una línea antes de volver a VERT5 para desplazar otra línea de pixels. Si la ha alcanzado, el fragmento de la rutina que comienza en CLREDG llena de ceros la fila de pixels inferior para que esa fila no aparezca en la pantalla.

La subrutina VLNSCR trata por separado los bytes de los márgenes, tal como vimos con HLNSCR. (Véase la ilustración de la derecha.) Para mover la parte central de la fila de pixels, la rutina incrementa HL y DE de modo que apunten el primer byte interior de la línea en curso y su correspondiente en la línea superior. Seguidamente calcula la longitud de la parte central (o sea, los bytes que pertenecen por entero al área de la ventana), carga esta información en BC y emplea la instrucción de movimiento en bloque LDIR para que toda esa parte central de la fila se desplace una línea hacia arriba.

Todas estas rutinas de desplazamiento son algo lentas. Esto se debe en parte a que están combinadas las de desplazamiento a izquierda y derecha y las de desplazamiento arriba o abajo, de forma que el programa debe realizar frecuentes comprobaciones con las que decide la parte de código máquina que ha de usar. En parte también se explica porque los bytes que están en los bordes izquierdo y derecho necesitan un tratamiento especial si se encuentran a caballo sobre esos bordes de la ventana.





Línea abajo
El desplazamiento horizontal presenta unos problemas determinados en los bordes de la ventana. Los bytes de estos márgenes deben ser "enmascarados" para separar los bits de pixels que caen dentro y fuera de la ventana, y además estos bytes deben desplazarse. Ambos procesos emplean el AND y el OR lógico y la instrucción SHIFT además de servirse de la pila para guardar el registro indicador de estado (PSR).
El desplazamiento vertical se simplifica mucho con el mapa de memoria de pantalla del Spectrum (véase p. 838). También se superpone una máscara al byte de pantalla para aislar los pixels de dentro y fuera de la ventana, además de la instrucción EX para intercambiar los contenidos de los registros DE y HL, que guardan punteros de direcciones de pantalla

Ventanas en el Spectrum

Instrucciones de operatoria

1) Entre el programa de demostración en BASIC
2) Haga SAVE "SCROLL" LINE 5
3) Digite el programa cargador de código máquina, y ejecútelo (RUN)
4) Haga SAVE "SCROLLMC" CODE 45312,410 directamente en cinta después del programa en BASIC
5) Rebobine la cinta y haga LOAD "SCROLL"

Listado assembly

```
10 PIXADR EQU  #22AA
20 WT     EQU  #B000
30 MASK1  EQU  #B002
40 MASK2  EQU  #B003
50 WNDWTB EQU  #B004
60 LEFTX  EQU  0
70 TOPY   EQU  1
80 RIGHTX EQU  2
90 BOTY   EQU  3
100 LBIT   EQU  4
110 RBIT   EQU  5
120 CURNTY EQU  6
130 DIR    EQU  7
140 LMASK  EQU  8
150 RMASK  EQU  9
160 LENGTH EQU  10
170        ORG  #B100
180 INIT   LD   HL,(WT)
190        PUSH HL
200        POP  IX
210        CALL INITW
220        RET
230 SCROLL LD   HL,(WT)
240        PUSH HL
250        POP  IX
260        BIT  1,(IX+DIR)
270        PUSH AF
280        CALL Z,HORIZ
290        POP  AF
300        CALL NZ,VERT
310        RET
320 HORIZ  LD   A,(IX+TOPY)
330        LD   (IX+CURNTY),A
340        BIT  0,(IX+DIR)
350        JR   Z,HORIZ1
360        LD   B,(IX+LMASK)
370        LD   A,(IX+RMASK)
380        JR   HORIZ2
390 HORIZ1 LD   B,(IX+RMASK)
400        LD   A,(IX+LMASK)
410 HORIZ2 LD   (MASK2),A
420        LD   A,B
430        LD   (MASK1),A
440 HORIZ3 LD   C,(IX+LEFTX)
450        LD   B,(IX+CURNTY)
460        CALL PIXADR
470        EX   DE,HL
480        LD   C,(IX+RIGHTX)
490        LD   B,(IX+CURNTY)
500        CALL PIXADR
510        BIT  0,(IX+DIR)
520        JR   Z,HORIZ4
530        EX   DE,HL
540 HORIZ4 CALL HLNSCR
550        LD   A,(IX+BOTY)
560        CP   (IX+CURNTY)
570        RET  Z
580        DEC  (IX+CURNTY)
590        JR   HORIZ3
600 HLNSCR LD   A,(MASK1)
610        LD   B,A
620        AND  (HL)
630        LD   C,A
640        LD   A,B
650        CPL
660        AND  (HL)
670        BIT  0,(IX+DIR)
680        JR   Z,HLN1
690        SRA  A
700        JR   HLN2
710 HLN1   SLA  A
720 HLN2   PUSH AF
730        OR   C
740        LD   (HL),A
750 NEXT   BIT  0,(IX+DIR)
760        JR   Z,HLN3
770        INC  HL
780        JR   HLN4
790 HLN3   DEC  HL
800 HLN4   LD   A,L
810        CP   E
820        JR   Z,LAST
830        POP  AF
840        BIT  0,(IX+DIR)
850        JR   Z,HLN5
860        RR   (HL)
870        JR   HLN6
880 HLN5   RL   (HL)
890 HLN6   PUSH AF
900        JR   NEXT
910 LAST   LD   C,(HL)
920        LD   A,(MASK2)
930        AND  C
940        LD   B,A
950        POP  AF
960        BIT  0,(IX+DIR)
970        JR   Z,HLN7
980        RR   C
990        JR   HLN8
1000 HLN7  RL   C
1010 HLN8  LD   A,(MASK2)
1020       CPL
1030       AND  C
1040       OR   B
1050       LD   (HL),A
1060       RET
1070 VERT  LD   C,(IX+LEFTX)
1080       BIT  0,(IX+DIR)
1090       JR   Z,VERT1
1100       LD   B,(IX+BOTY)
1110       JR   VERT2
1120 VERT1 LD   B,(IX+TOPY)
1130 VERT2 LD   (IX+CURNTY),B
1140       CALL PIXADR
1150       PUSH HL
1160       PUSH HL
1170       LD   C,(IX+RIGHTX)
1180       LD   B,(IX+CURNTY)
1190       CALL PIXADR
1200       POP  DE
1210       AND  A
1220       SBC  HL,DE
1230       LD   A,L
1240       DEC  A
1250       LD   (IX+LENGTH),A
1260 VERT3 BIT  0,(IX+DIR)
1270       JR   Z,VERT4
1280       INC  (IX+CURNTY)
1290       JR   VERT5
1300 VERT4 DEC  (IX+CURNTY)
1310 VERT5 LD   C,(IX+LEFTX)
1320       LD   B,(IX+CURNTY)
1330       CALL PIXADR
1340       POP  DE
1350       PUSH HL
1360       CALL VLNSCR
1370       LD   A,(IX+CURNTY)
1380       BIT  0,(IX+DIR)
1390       JR   Z,VERT6
1400       CP   (IX+TOPY)
1410       JR   VERT7
1420 VERT6 CP   (IX+BOTY)
1430 VERT7 JR   NZ,VERT3
1440 CLREDG POP HL
1450       LD   A,(IX+LMASK)
1460       AND  (HL)
1470       LD   (HL),A
1480       LD   B,(IX+LENGTH)
1490       LD   A,0
1500 CLR1  INC  HL
1510       LD   (HL),A
1520       DJNZ CLR1
1530       INC  HL
1540       LD   A,(IX+RMASK)
1550       AND  (HL)
1560       LD   (HL),A
1570       RET
1580 VLNSCR LD  A,(IX+LMASK)
1590       CALL ENDBYT
1600       INC  HL
1610       INC  DE
1620       LD   B,0
1630       LD   C,(IX+LENGTH)
1640       LDIR
1650       LD   A,(IX+RMASK)
1660 ENDBYT LD  B,A
1670       CPL
1680       AND  (HL)
1690       LD   C,A
1700       EX   DE,HL
1710       LD   A,B
1720       AND  (HL)
1730       OR   C
1740       LD   (HL),A
1750       EX   DE,HL
1760       RET
1770 INITW LD   A,(IX+RIGHTX)
1780       CP   (IX+LEFTX)
1790       JR   Z,ERROR
1800       JR   C,ERROR
1810       LD   A,(IX+TOPY)
1820       CP   (IX+BOTY)
1830       JR   Z,ERROR
1840       JR   C,ERROR
1850       LD   C,(IX+LEFTX)
1860       LD   B,(IX+TOPY)
1870       CALL PIXADR
1880       PUSH HL
1890       LD   (IX+LBIT),A
1900       LD   C,(IX+RIGHTX)
1910       LD   B,(IX+TOPY)
1920       CALL PIXADR
1930       LD   (IX+RBIT),A
1940       POP  BC
1950       LD   A,C
1960       CP   L
1970       JR   Z,ERROR
1980 LFTMSK LD  B,(IX+LBIT)
1990       LD   A,0
2000 L1    SCF
2010       RRA
2020       DJNZ L1
2030       LD   (IX+LMASK),A
2040 RTMASK LD  B,(IX+RBIT)
2050       LD   A,255
2060 L2    AND  A
2070       RRA
2080       DJNZ L2
2090       LD   (IX+RMASK),A
2100       RET
2110 ERROR RST  8
2120       DEFB 25
```

Programa de demostración en BASIC

```
5 CLEAR 32767
10 LOAD ""CODE
20 LET WT=45056
30 LET WINDOWTABLE=45060: REM B004 HEX
40 LET LEFTX=0
50 LET TOPY=1
60 LET RIGHTX=2
70 LET BOTY=3
75 LET DIR=7
80 LET SCROLL=45322
90 LET INIT=45312
100 BORDER 6
110 PAPER 4: INK 2
120 CLS
180 POKE WT,4: POKE WT+1,176
190 REM WT & WT+1 NOW CONTAIN ADDRESS 45060
IN LO,HI FORMAT
200 POKE WINDOWTABLE+LEFTX,5
210 POKE WINDOWTABLE+TOPY,80
220 POKE WINDOWTABLE+RIGHTX,250
230 POKE WINDOWTABLE+BOTY,35
240 RANDOMIZE USR INIT
250 FOR Y=0 TO 175
260 PLOT 0,Y
270 DRAW 255,0
280 NEXT Y
290 POKE WINDOWTABLE+DIR,2
300 FOR Y=0 TO 45
310 RANDOMIZE USR SCROLL
320 NEXT Y
400 PRINT AT 12,20;"HELLO";
410 POKE WINDOWTABLE+DIR,0
420 FOR I=1 TO 130
430 RANDOMIZE USR SCROLL
440 NEXT I
470 POKE WINDOWTABLE+DIR,3
480 FOR I=1 TO 35
490 RANDOMIZE USR SCROLL
500 NEXT I
510 POKE WINDOWTABLE+DIR,1
520 FOR I=1 TO 130
530 RANDOMIZE USR SCROLL
540 NEXT I
550 POKE WINDOWTABLE+DIR,2
560 FOR I=1 TO 35
570 RANDOMIZE USR SCROLL
580 NEXT I
590 GO TO 410
999 STOP
```

Cargador de código máquina

```
100 LET a=45312
110 FOR l=1000 TO 1500 STEP 10
120 LET s=0
130 READ b
140 POKE a,b
150 LET s=s+b
160 NEXT a
170 READ b
180 IF s<>b THEN  PRINT "ERROR IN LINE ";L: STOP
190 NEXT l
1000 DATA 42,0,176,229,221,225,205,69,1167
1010 DATA 178,201,42,0,176,229,221,225,1272
1020 DATA 221,203,7,78,245,204,29,177,1164
1030 DATA 241,196,184,177,201,221,126,1,1347
1040 DATA 221,119,6,221,203,7,70,40,887
1050 DATA 8,221,70,8,221,126,9,24,687
1060 DATA 6,221,70,9,221,126,8,50,711
1070 DATA 3,176,120,50,2,176,221,78,826
1080 DATA 0,221,70,6,205,170,34,235,941
1090 DATA 221,78,2,221,70,6,205,170,973
1100 DATA 34,221,203,7,70,40,1,235,811
1110 DATA 205,103,177,221,126,3,221,190,1246
1120 DATA 6,200,221,53,6,24,215,58,783
1130 DATA 2,176,71,166,79,120,47,166,827
1140 DATA 221,203,7,70,40,4,203,47,795
1150 DATA 24,2,203,39,245,177,119,221,1030
1160 DATA 203,7,70,40,3,35,24,1,383
1170 DATA 43,125,187,40,16,241,221,203,1076
1180 DATA 7,70,40,4,203,30,24,2,380
1190 DATA 203,22,245,24,226,78,58,3,859
1200 DATA 176,161,71,241,221,203,7,70,1150
1210 DATA 40,4,203,25,24,2,203,17,518
1220 DATA 58,3,176,47,161,176,119,201,941
1230 DATA 221,78,0,221,203,7,70,40,840
1240 DATA 5,221,70,3,24,3,221,70,617
1250 DATA 1,221,112,6,205,170,34,229,978
1260 DATA 229,221,78,2,221,70,6,205,1032
1270 DATA 170,34,209,167,237,82,125,61,1085
1280 DATA 221,119,10,221,203,7,70,40,891
1290 DATA 5,221,52,6,24,3,221,53,585
1300 DATA 6,221,78,0,221,70,6,205,807
1310 DATA 170,34,209,229,205,40,178,221,1286
1320 DATA 126,6,221,203,7,70,40,5,678
1330 DATA 221,190,1,24,3,221,190,3,853
1340 DATA 32,209,225,221,126,8,166,119,1106
1350 DATA 221,70,10,62,0,35,119,16,533
1360 DATA 252,35,221,126,9,166,119,201,1129
1370 DATA 221,126,8,205,58,178,35,19,850
1380 DATA 6,0,221,78,10,237,176,221,949
1390 DATA 126,9,71,47,166,79,235,120,853
1400 DATA 166,177,119,235,201,221,126,2,1247
1410 DATA 221,190,0,40,67,56,65,221,860
1420 DATA 126,1,221,190,3,40,57,56,694
1430 DATA 55,221,78,0,221,70,1,205,851
1440 DATA 170,34,229,221,119,4,221,78,1076
1450 DATA 2,221,70,1,205,170,34,221,924
1460 DATA 119,5,193,121,189,40,25,221,913
1470 DATA 70,4,62,0,55,31,16,252,490
1480 DATA 221,119,8,221,70,5,62,255,961
1490 DATA 167,31,16,252,221,119,9,201,1016
1500 DATA 207,25,0,0,0,0,0,0,232
```



Ordenando los naipes

El juego “Invertir” tiene como objetivo acomodar una lista de números por orden ascendente con el menor número de movimientos

El programa genera al azar una lista de números para clasificar. Sólo se puede cambiar el orden de los números invirtiendo grupos específicos dentro de la lista. Por ejemplo, si el ordenador genera la siguiente lista al azar en respuesta a la solicitud de nueve números hecha por el jugador:

2 8 4 7 1 5 6 9 3

y el jugador especifica luego “Invertir? 5”, se invertirán los primeros 5 números y la lista quedará en:

1 7 4 8 2 5 6 9 3

Resolver un puzzle como éste no le debería llevar ni mucho tiempo ni mucho esfuerzo, y en principio parece tarea fácil hallar un algoritmo que lo resuelva fácilmente. Sin embargo, en la práctica es difícil definir uno que sea realmente bueno. Supongamos que en la lista hay n números. El algoritmo más obvio es éste:

- Hallar el número mayor de la lista e invertir todos los números hasta su posición. El número mayor está ahora en el extremo izquierdo de la lista.
- Invertir todos los n números, de modo que el número mayor quede en su posición deseada, en el extremo derecho de la lista. Esto sólo ha llevado dos inversiones.
- Hallar el segundo número mayor y volver a repetir todo el proceso. Desplazar este número hasta su posición deseada exige un movimiento “Invertir n-1”.
- Repetir el procedimiento hasta conseguir la ordenación completa.

Este algoritmo resuelve *siempre* el puzzle en 2n-3 movimientos. Pero se puede obtener una solución que lleve menos movimientos que ésta. Para demostrar cómo una estrategia directa puede reducir la cantidad de movimientos, consideremos el ejemplo que damos en el recuadro. Nuestro algoritmo llevaría siete (2×5−3) movimientos, pero un jugador hábil lo podría hacer en cuatro.

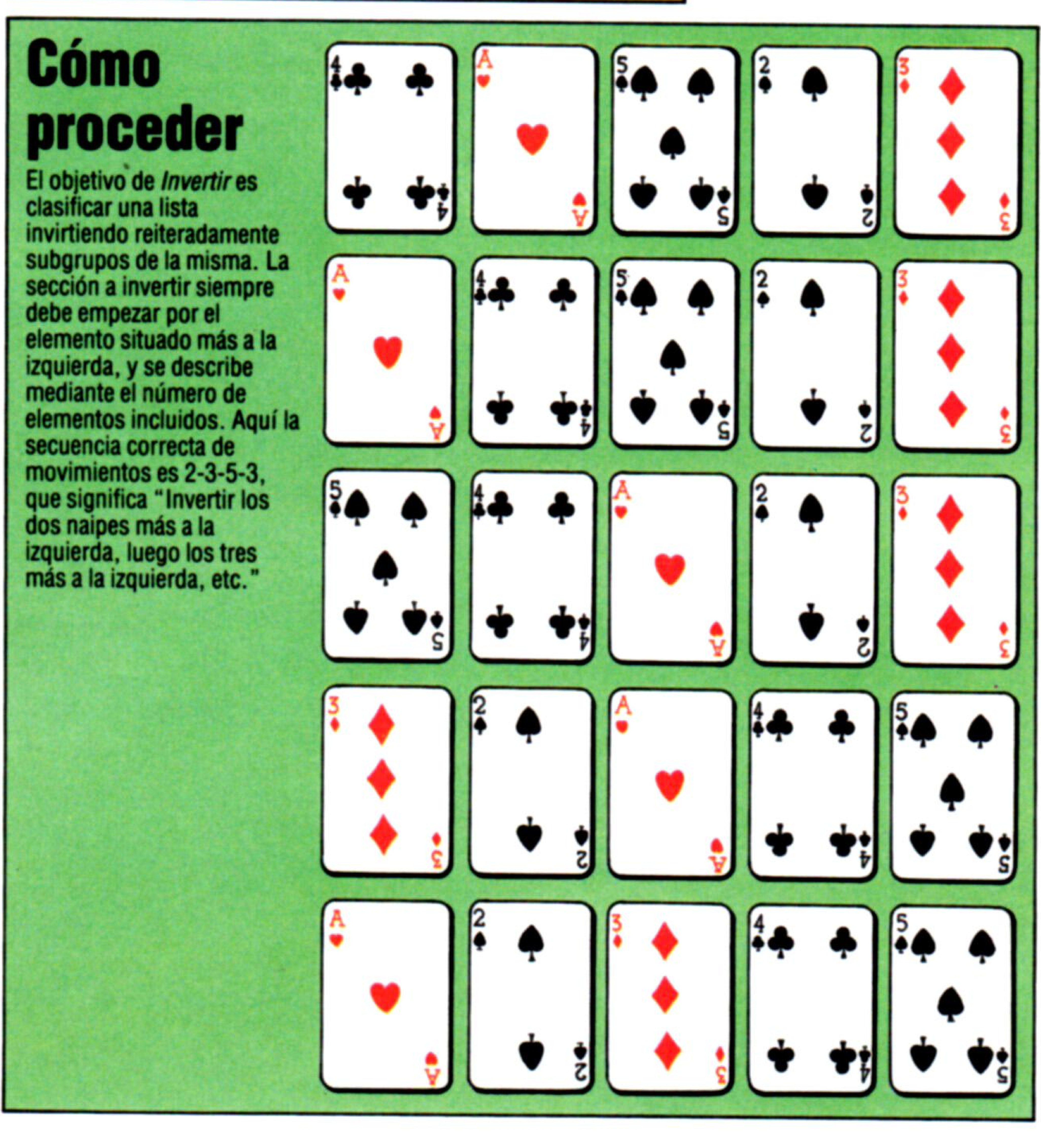
Este programa es un ejemplo sencillo de toda una serie de juegos de inversión que la gente ha creado e investigado. Quizá a usted le agradará tratar de desarrollar juegos para invertir desde cualquiera de los extremos de la línea, o en los que hubiera de clasificar no sólo una línea de números sino una cuadrícula. Si diseñara su propia versión del juego, tal vez deseara darle más vida utilizando bloques de diferentes colores para reemplazar los números. El objetivo del juego podría ser, entonces, reacomodar una línea de bloques de acuerdo a unos bloques de colores situados en la parte superior de la pantalla. Tal vez también le interesará tratar de incorporarle al programa un algoritmo que les fuera de ayuda a los jugadores que se quedaran encallados.

Invertir

```
20 DIM a(20)
30 CLS : PRINT "Invertir!"
40 INPUT "Cuantos numeros?";n
50 IF n < 0 OR n > 20 OR n <> INT n THEN GO TO 30
60 REM Mezclar la lista
70 FOR i=1 TO n:LET a(i)=i: NEXT i
80 RANDOMIZE
90 FOR i=1 TO n
100 LET r=INT (RND*n+1)
110 LET x=a(r): LET a(r)=a(i): LET a(i)=x
120 NEXT i
130 LET t=1
135 REM Imprimir el tablero
140 CLS: PRINT "Movimiento";t;": La lista es:": PRINT
150 FOR i=1 TO n: PRINT a(i);"   ";: NEXT i
152 REM Verificar si se gana
154 LET i=1
156 IF a(i)=i THEN LET i=i+1: IF i < =n THEN GO
      TO 156
158 IF i > n THEN GO TO 230
159 REM Hacer una pasada
160 PRINT: PRINT: INPUT "Invertir?";r
170 IF r <> INT r OR r < 0 OR r > n THEN GO TO 140
175 REM Invertir r
180 LET t=t+1
190 FOR i=1 TO INT (r/2)
200 LET x=a(i): LET a(i)=a(r-i+1): LET a(r-1+1)=x
210 NEXT i
220 GO TO 140
230 REM Un ganador!
240 PRINT: PRINT: PRINT "Ha terminado en ";t;
    "movimientos"
250 PRINT: INPUT "Juega otra vez (s/n) ?";A$
260 IF a$="S" OR a$="s" THEN RUN
270 CLS: STOP
```

Complementos al BASIC

En el Commodore 64 y el Vic-20, reemplazar RANDOMIZE por XX=RND(−TI), reemplazar RND*N por RND(1)*N, y reemplazar CLS por PRINT CHR$(147)
En el BBC Micro eliminar la línea 80, y reemplazar R=INT(RND*N+1) por R=RND(N)
En el Oric-1 y el Oric Atmos eliminar la línea 80 y reemplazar RND*N por RND(1)*N



Cómo proceder

El objetivo de *Invertir* es clasificar una lista invirtiendo reiteradamente subgrupos de la misma. La sección a invertir siempre debe empezar por el elemento situado más a la izquierda, y se describe mediante el número de elementos incluidos. Aquí la secuencia correcta de movimientos es 2-3-5-3, que significa “Invertir los dos naipes más a la izquierda, luego los tres más a la izquierda, etc.”

De ámbito mundial

Sharp Corporation fabrica una enorme gama de productos electrónicos, desde transistores a ordenadores y robots industriales

Sharp Corporation siempre ha sido partícipe de las mayores innovaciones tecnológicas. Pero el primer producto suyo que alcanzó el éxito fue un artículo sumamente humilde: el lapicero Ever Sharp ("siempre agudo"). Su creador, Tokuji Hayakawa, creó la Sharp en 1915 para fabricar su invento; y en los años posteriores la empresa fue creciendo de forma continua. En 1925 se inició en la electrónica con un receptor de radio de cristal; y su introducción en los mercados internacionales de la electrónica de consumo se produjo en los años de la posguerra, cuando comenzó a producir aparatos de televisión y otros electrodomésticos. A mediados de los años sesenta la empresa intervino en el mercado de maquinaria de gestión con una serie de calculadoras de sobremesa. En la actualidad es una inmensa multinacional, subdividida en seis grupos de fabricación, con 34 plantas de producción en 30 países, sin contar Japón.

El primer ordenador Sharp que se comercializó en Gran Bretaña fue el MZ80K, que se lanzó en 1981. Al año siguiente, la empresa agregó a su gama el MZ380A y el MZ380B. A pesar de que estos ordenadores se comercializaron como máquinas de oficina, también fueron objeto de una buena acogida por parte de los usuarios de micros personales. Cada modelo viene equipado con una pantalla incorporada y unidades de cassette. Al principio, estos ordenadores se comercializaron como "máquinas limpias", haciendo hincapié en la ausencia de un lenguaje residente en ROM. Este hecho representaba una ventaja: se podían cargar, desde cassette, numerosos lenguajes, incluyendo CP/M.



Grupo de Instrumentos Industriales (IIG)
Los aspectos tecnológicos que son objeto de investigación en el Centro de Ingeniería de Sharp se trasladan a la planta IIG (que vemos en la fotografía) de Yamato-Koriyama-shi, en Nara (Japón), donde se diseñan los productos de la empresa (incluyendo calculadoras y ordenadores personales)

A principios de 1983 Sharp empezó a vender en Gran Bretaña una gama completa de ordenadores personales y de oficina; en esa fecha, además, la lista de productos de la empresa se amplió para incluir la máquina de gestión MZ-3541 y el ordenador de bolsillo PC-1500. El éxito de este último condujo a la comercialización del ordenador de bolsillo PC-1251.

La empresa penetró en el mercado del ordenador personal con el lanzamiento del Sharp MZ-711. Ésta es la versión europea de la serie japonesa MZ-700, y el inmenso juego de caracteres japoneses del original ha permitido un espacio extra para facilidades de gráficos en el modelo europeo. La máquina lleva instalada como estándar una grabadora de datos y se incluye espacio para una impresora-plotter opcional.

En mayo de 1984 la empresa puso a la venta el PC-1500A, una versión mejorada del PC-1500. El nuevo modelo posee 8,9 Kbytes de RAM, que se pueden ampliar a 24 Kbytes. Antes de que finalice 1984 la empresa tiene planeado lanzar la máquina de bolsillo PC-1350, con una visualización de cuatro líneas y capacidades para gráficos. La Sharp Corporation también tiene la intención de introducir el paquete Sharpwriter, que es una fusión de la máquina de escribir eléctrica Sharp ZX-401 y el microordenador MZ-3541. La máquina de escribir se utiliza como un teclado-impresora conectado al ordenador mediante una interface RS232.

Cuando se le pregunta acerca de los futuros desarrollos de Sharp, el director de ventas Rod Goodier afirma que a él "siempre le ha interesado ampliar el mercado del ordenador de bolsillo, que ofrece un potencial tan grande". Ello no significa que se deje de lado el mercado del ordenador personal. "Con la familia MZ-700 realmente no escatimaremos gastos y estamos decididos a mantener una posición sólida en el mercado de ordenadores personales." Peter Fletcher, portavoz de la empresa, explica que "la división de equipos para oficina, que incluye los ordenadores personales, es una innovación relativamente reciente, y representa el 25 % del movimiento total en Gran Bretaña. El plan consiste en aumentar la proporción de movimientos".

En la actualidad Sharp de Gran Bretaña sólo posee un local para almacenaje y comercialización. La empresa está construyendo una fábrica en Wrexham (Gales), donde se ensamblarán grabadoras de video. Con el plan se espera producir 60 000 máquinas en 1985 para distribuirlas a toda Europa.

Al preguntársele acerca de la participación de Sharp con otras empresas en la invasión de MSX que se anunciaba para el otoño de 1984, Rod Goodier replicó que "Sharp ha desarrollado un sistema MSX, pero por el momento no tenemos planeado lanzarlo en Gran Bretaña".

mi computer

Editorial Delta, S.A.



10044
9 788485 822836